Supplement to: Measurement and comparison of distributional shift with applications to ecology, socioeconomics, and image analysis

Kenneth J. Locey[1*], Brian D. Stein[1]

[1] Center for Quality, Safety and Value Analytics, Rush University Medical Center, Chicago, Illinois, 60612, USA.

[*] Corresponding author. email: Kenneth_J_Locey@rush.edu

Appendix 1. Analysis of species abundance distributions and the use of distributional shift (DS) as a measure of species rarity.

These analyses were conducted using a Python (v3.8.12) Jupyter Notebook, which is available in a public GitHub repository hosted by our institution: https://github.com/Rush-Quality-Analytics/distributional_shift/blob/main/RDS-main.ipynb.

We used a previously published data compilation of nearly 40,100 samples from geographically unique sites of ecological communities of plants, animals, fungi, archaea, and bacteria (Lennon and Locey 2020). The data are publicly available, compiled from large-scale sampling projects, and were primarily used for the analysis of ecological scaling laws and the prediction of global biodiversity (Locey and Lennon 2016, Louca *et al*. 2019, Lennon and Locey 2020). Each sample is represented as a vector of abundances, which can then be analyzed as a species-abundance distribution (SAD).

We calculated distributional shift (DS) for each sample having 10 or more species. In general, SADs with less than 10 species are not considered sufficient for analysis (Locey and Lennon 2016). Importantly, we calculated DS based on SADs with $\log_2$ abundance classes as opposed to arithmetic abundance classes. The use of logarithmic abundance classes is a common practice in ecological studies and offered greater computational efficiency, e.g., as an SAD with a maximum species abundance of, e.g., 10^4, would otherwise have that many discrete abundance classes, the vast majority of which may be empty. However, on the log2 scale there would only be 14 discrete abundance classes.

The python functions for calculating DS were as follows, where NSECF stands for normalized sum of exponentiated cumulative frequencies:

```
def NSECF(p):
    p_bins = len(p)
    p_obs = sum(p)

    z_p = (p_bins + 1)/p_bins
    p = [sum(p[:ii+1])**(z_p) for ii in range(len(p))]
    return np.sum(np.array(p)/(p_obs**z_p)) - 1
```

```
# 5. Distributional shift (DS)
# Convert the abundances to logarithmic scale (base 2)
abundances = np.log2(RAD)

# Define the bins for the histogram
min_abundance = 0 #np.floor(min(abundances))
max_abundance = np.ceil(max(abundances))
bins = np.arange(min_abundance, max_abundance + 1, 1)

# Compute the histogram
hist, bin_edges = np.histogram(abundances, bins=bins)

# Use the right side of the bin edges as bin values
bin_values = bin_edges[1:]

# Convert histogram to list
bin_heights = hist.tolist()
ds = DS(bin_heights)

# 6. Normalized sums of exponentiated cumulative frequencies
nsecf = NSECF(bin_heights)
```

We also calculated the studies' original measure of species rarity (R), i.e., log-modulo skewness:

$$R = \log_{10}(|\gamma| + 1), \text{ if } 0 \leq \gamma$$

$$R = -\log_{10}(|\gamma| + 1), \text{ if } \gamma < 0$$

This measure was computed via the following python code:

```
# 2. log-modulo transformation of skewnness
lms = np.log10(np.abs(float(skewness)) + 1)
if skewness < 0:
    lms = lms * -1
log_mod_skew = float(lms)
```

Appendix 2. Analysis of US poverty rates, family incomes, and the use of distributional shift (DS) as a measure of poverty.

These analyses were conducted using a Python (v3.8.12) Jupyter Notebook, which is available in a public GitHub repository hosted by our institution: https://github.com/Rush-Quality-Analytics/distributional_shift/blob/main/RDS-main.ipynb.

We examined distributional shift (DS) as a measure of poverty using data from the US Census Bureau on family incomes and poverty rates, from 2010 to 2022 [25, 26]. Family income data included 16 bins of inflation-corrected annual family incomes (Table A1).

Table A1. 16 bins of inflation-corrected annual family incomes. These bins are not of equal width. Results are given for the year 2022, which included data from 81,432,908 families.

Income bin	No. of families included in 2022	% of total in 2022
$0 to $10,000	2,503,187	3.07392559
$10,000 to $14,999	1,498,443	1.84009516
$15,000 to $19,999	1,600,812	1.96580478
$20,000 to $24,999	2,028,300	2.49076208
$25,000 to $29,999	2,236,899	2.74692265
$30,000 to $34,999	2,386,079	2.93011640
$35,000 to $39,999	2,463,465	3.02514678
$40,000 to $44,999	2,525,965	3.10189709
$45,000 to $49,999	2,616,615	3.21321572
$50,000 to $59,999	5,242,942	6.43835782
$60,000 to $74,999	7,445,194	9.14273378
$75,000 to $99,999	11,250,047	13.81511145
$100,000 to $124,999	9,364,651	11.49983615
$125,000 to $149,999	7,122,680	8.74668506
$150,000 to $199,999	9,059,762	11.12543101
$200,000 or greater	12,087,867	14.84395842

Appendix 3. Analysis of distributions for the production of food commodities among nations in 2022.

These analyses were conducted using a Python (v3.8.12) Jupyter Notebook, which is available in a public GitHub repository hosted by our institution: https://github.com/Rush-Quality-Analytics/distributional_shift/blob/main/RDS-main.ipynb.

We examined distributional shift (DS) as a measure of scarcity with respect to the production of food commodities (e.g., wheat, rice, chicken meat) among nations. Food production was reported in numbers of metric tons. We calculated DS and the Gini coefficient of inequality with respect to each food commodity. The Gini coefficient is the most commonly used measure of economic inequality and is equal to the area under the Lorenz curve, i.e., a plot of cumulative wealth (e.g., income) versus the cumulative portion of a sample (e.g., set of nations) (Williams and Doessel 2006, Houghton and Khandker 2009):

$$Gini = 1 - \sum_{i=1}^{N} (x_i - x_{i-1})\,(y_i + y_{i-1})$$

The Gini coefficient reflects a transformation of the sample variance, bounded between 0 (all values are equal) and 1 (one non-zero observation and an infinite number of 0-valued observations). We calculated the Gini coefficient using a square root transformation on observed values, as it produced a strong linear relationship to DS. The python function for calculating the Gini coefficient is given below:

```
def gini_coefficient(x):
    """
    Compute Gini coefficient of array of values
    From: https://stackoverflow.com/questions/39512260/calculating-gini-coefficient-in-python-numpy
    """
    diffsum = 0
    for i, xi in enumerate(x[:-1], 1):
        diffsum += np.sum(np.abs(xi - x[i:]))
    return diffsum / (len(x)**2 * np.mean(x))
```

Appendix 4. Methodological details on relationships of RDS to comparative measures

These analyses were conducted using a Python (v3.8.12) Jupyter Notebook, which is available in a public GitHub repository hosted by our institution: https://github.com/Rush-Quality-Analytics/distributional_shift/blob/main/RDS_compare_measures.ipynb.

We based our analysis on random samples generated via five models (i.e., Poisson, Gaussian, negative binomial, lognormal, and Weibull) and seven (n, k) pairs: {(10, 3), (1000, 3), (10, 50), (100, 50), (100, 5), (100, 500), (100, 500)}, where n is the number of total observations and k is the number of bins. These operations were conducted via a simplistic function, generate_distributions (below), that uses the python-based numerical computing library Numpy (np) to sample n observations from a specific statistical model using the `np.random` module. The resulting observations were then binned into k equally spaced bins using the `np.histogram` function. A screenshot of the generate_distributions function is provided below:

```python
def generate_distributions(N, n, k, model, **kwargs):
    dists = []
    while len(dists) < N:

        if model == 'Poisson':
            lambda_param = kwargs.get('lambda_p', 5)  # Default lambda value is 5
            data = np.random.poisson(lambda_param, n)

        elif model == 'Gaussian':
            mean = kwargs.get('mean', 0)  # Default mean is 0
            std = kwargs.get('std', 1)  # Default standard deviation is 1
            data = np.random.normal(mean, std, n)

        elif model == 'Negative binomial':
            r = kwargs.get('r', 1)  # Number of successes
            p = kwargs.get('p', 0.5)  # Probability of success in each trial
            data = np.random.negative_binomial(r, p, n)

        elif model == 'Lognormal':
            mean = kwargs.get('mean', 0)  # Mean of the underlying normal distribution
            sigma = kwargs.get('sigma', 1)  # Standard deviation of the underlying normal distribution
            data = np.random.lognormal(mean, sigma, n)

        elif model == 'Weibull':
            a = kwargs.get('a', 1)  # Shape parameter
            data = np.random.weibull(a, n)

        else:
            raise ValueError("Unsupported model type")

        hist_vals, bins = np.histogram(data, bins=k, density=False)
        dists.append(hist_vals.tolist())

    return dists
```

We then examined relationships of Relative Distributional Shift (RDS) to Chi-square distance (CSD), Kullback-Leibler divergence (KLD), Kolmogorov-Smirnov distance (KSD), Earth Mover's distance (EMD), Ranked Probability Score (RPS), and histogram *non*-intersection (1 – HI) (see Table 1 in manuscript). Screenshots of python source code are given below.

Relative Distributional Shift (RDS):

```
def RDS(p, q):

    # Calculate Relative distribution shift

    # q is the reference distribution
    # p is the query distribution
    p_bins = len(p)
    p_obs = sum(p)

    q_bins = len(q)
    q_obs = sum(q)

    z_p = (p_bins + 1)/p_bins
    p = [sum(p[:ii+1])**(z_p) for ii in range(len(p))]
    Sp = np.sum(np.array(p)/(p_obs**z_p)) - 1
    Sp = Sp/(p_bins - 1)

    z_q = (q_bins + 1)/q_bins
    q = [sum(q[:ii+1])**(z_q) for ii in range(len(q))]
    Sq = np.sum(np.array(q)/(q_obs**z_q)) - 1
    Sq = Sq/(q_bins - 1)

    return Sq - Sp
```

Kolmogorov-Smirnov Distance (KSD):

```
def kolmogorov_smirnov_distance(p, q):

    # Calculate KS distance

    # q is the reference distribution
    # p is the query distribution

    # Ensure both lists are numpy arrays with dtype=float
    p = np.array(p, dtype=float)
    q = np.array(q, dtype=float)

    # Normalize the distributions to ensure they sum to 1
    p /= p.sum()
    q /= q.sum()

    # Calculate cumulative distributions
    P = np.cumsum(p)
    Q = np.cumsum(q)

    # Calculate the KS distance
    ks_distance = np.max(np.abs(P - Q))

    return ks_distance
```

Earth Mover's Distance (EMD); uses the `linear_sum_assignment` function from the `optimize` module of the `Scipy` scientific computing library.

```
def earth_movers_distance(p, q):

    # Calculate Earth Mover's Distance

    # q is the reference distribution
    # p is the query distribution

    # Ensure both lists are numpy arrays with dtype=float
    p = np.array(p, dtype=float)
    q = np.array(q, dtype=float)

    # Normalize the distributions to ensure they sum to 1
    p /= p.sum()
    q /= q.sum()

    # Calculate cumulative distributions
    P = np.cumsum(p)
    Q = np.cumsum(q)

    # Calculate the cost matrix
    C = np.abs(np.subtract.outer(P, Q))

    # Solve the linear sum assignment problem
    row_ind, col_ind = linear_sum_assignment(C)

    # Calculate the Earth Mover's Distance
    emd = C[row_ind, col_ind].sum()

    return emd
```

Kullback-Leibler divergence (KLD)

```
def kl_divergence(p, q):

    # Calculate Kullback-Leibler Divergence

    # q is the reference distribution
    # p is the query distribution

    # Ensure both lists are numpy arrays with dtype=float

    p = np.array(p, dtype=float)
    q = np.array(q, dtype=float)

    p /= p.sum()
    q /= q.sum()

    kl_div = np.sum(p * np.log(p / q))
    return kl_div
```

Chi-square distance (CSD):

```
def chi_square_distance(p, q):

    # Calculate chi-square distance

    # q is the reference distribution
    # p is the query distribution

    p = np.array(p)/np.sum(p)
    q = np.array(q)/np.sum(q)

    return np.sum(((p - q)**2 / (p + q))) / 2
```

Histogram intersection (the values returned were subtracted from 1 to obtain non-intersection):

```
def histogram_intersection(p, q):

    # Calculate histogram intersection

    # q is the reference distribution
    # p is the query distribution

    minima = np.minimum(p, q)
    hi = np.true_divide(np.sum(minima), np.sum(p))

    if hi > 1 or hi < 0:
        print('Error, HI =', hi)
        print(p)
        print(q)
        return
    return hi
```

Ranked Probability Score (RPS):

```
def rank_probability_score(p, q):

    # Calculate the ranked probability score

    # q is the reference distribution
    # p is the query distribution

    # Ensure both lists are numpy arrays with dtype=float
    p = np.array(p, dtype=float)
    q = np.array(q, dtype=float)

    # Normalize the distributions to ensure they sum to 1
    p /= p.sum()
    q /= q.sum()

    # Calculate cumulative distributions
    P = np.cumsum(p)
    Q = np.cumsum(q)

    # Calculate Rank Probability Score
    rps = np.sum((P - Q)**2)

    return rps
```

After calculating the above measures for random samples from each model and each combination of *n* and *k*, we used the `linregress` function of the `stats` module from the `scipy` library to perform ordinary least regressions using the absolute value of RDS.

Appendix 5. Methodological details on the application of RDS to image analysis

These analyses were conducted using two Python (v3.8.12) Jupyter Notebooks, which are available in a public GitHub repository hosted by our institution:

1) To generate Figure 6 of the main manuscript: https://github.com/Rush-Quality-Analytics/distributional_shift/blob/main/main_image_analysis.ipynb
2) To generate Supplemental Figures 7-57: https://github.com/Rush-Quality-Analytics/distributional_shift/blob/main/other_image_analysis.ipynb

Each of 50 videos were processed frame-by-frame. Each frame contained three two-dimensional RGB pixel vectors from which R, G, and B histograms were formed. These histograms were then concatenated in R, G, B order (*sensu* Cassisi *et al*. 2012); an otherwise common operation. These operations were performed via use of the python version of the OpenCV computer vision library using the following python function:

```
def flatten_histogram(image, num_bins, htype='color'):

    if htype == 'color':
        """
        Construct a flattened 1D histogram for a color image by concatenating the histograms
        of each color channel.
        """
        if len(image.shape) == 3:  # Check if the image is indeed in color (RGB)
            channels = cv2.split(image)
            histograms = []
            for chan in channels:
                hist, _ = np.histogram(chan, bins=num_bins, range=(0, num_bins))
                histograms.append(hist)
            histogram = np.concatenate(histograms) / np.sum(histograms)
        else:  # For grayscale images, compute histogram directly
            histogram, _ = np.histogram(image, bins=num_bins, range=(0, num_bins))
            histogram = histogram / np.sum(histogram)
        return histogram
```

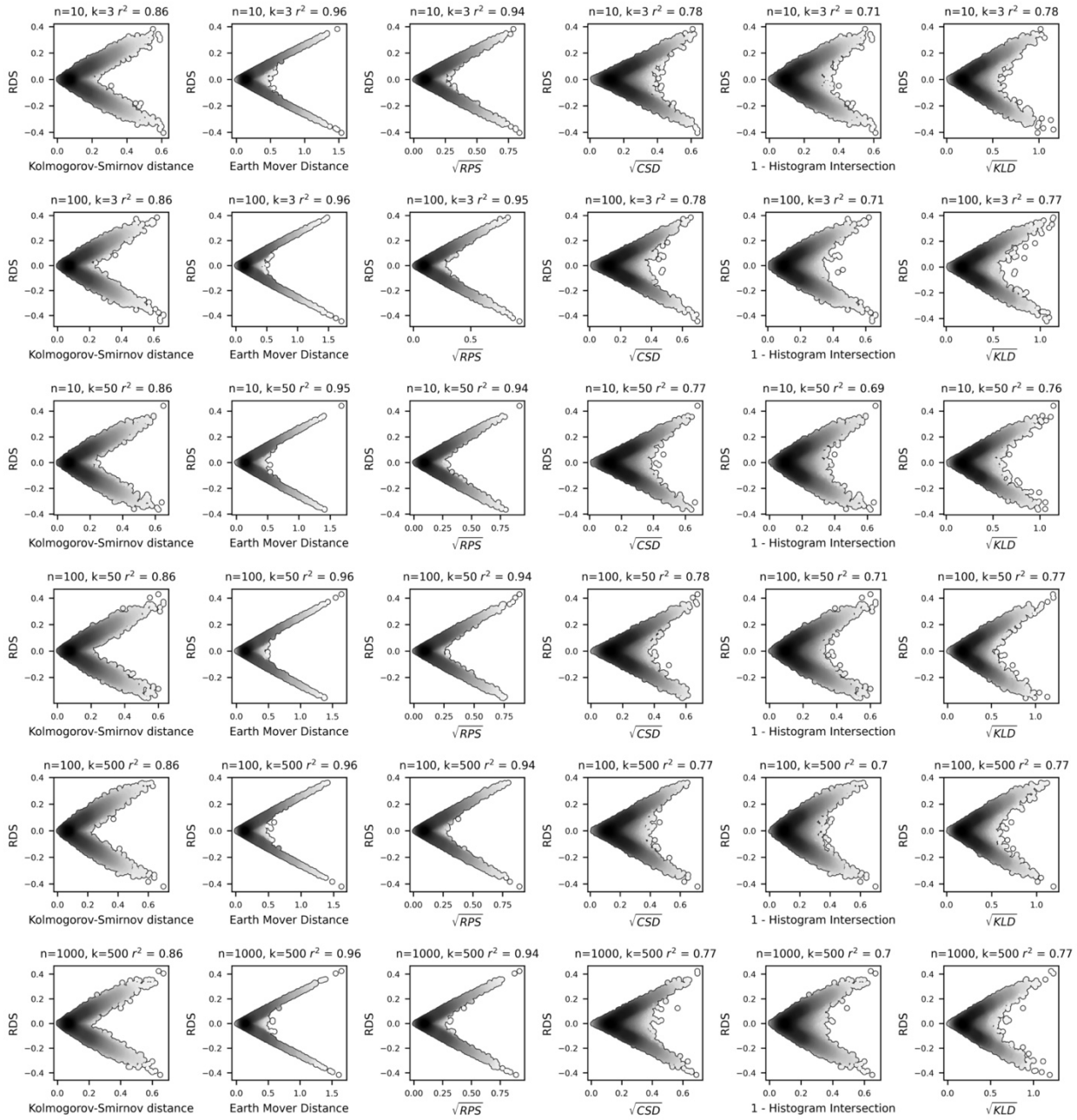
The flatten_histogram function allows users to produce histograms based on color hue and intensity. However, we did not analyze those properties in our work and, instead, focusing on comparisons of histograms based on RGB color values.

Several videos in our analyses began with a fade from black and/or ended with a fade to black. To remove this effect, our analysis ignored the first and last 10 frames of those videos:

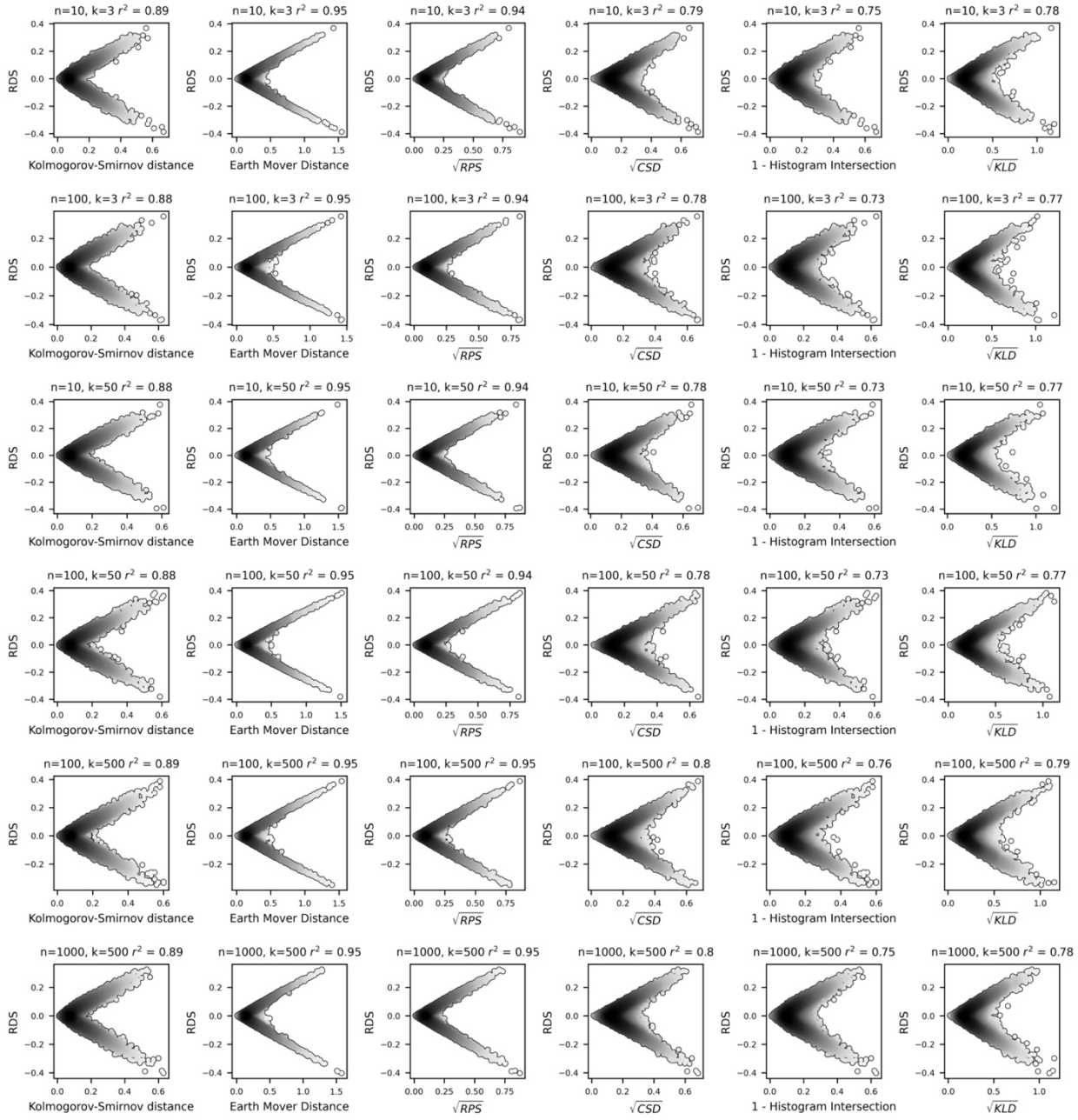
```
def process_video_with_metric(video_path, metric_name, num_bins, return_frames):
    video = cv2.VideoCapture(video_path)
    total_frames = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
    metric_values = []
    frame_indices = []
    first_frame = None
    max_rds_value = None
    max_rds_frame = None

    # Define videos that need trimming
    videos_to_trim = ['data/time_lapse_video/biology/855852-sd_640_360_30fps.mp4',
                      'data/time_lapse_video/biology/856030-sd_640_360_25fps.mp4',
                      'data/time_lapse_video/biology/856006-sd_640_360_25fps.mp4',
                      'data/time_lapse_video/biology/856990-sd_640_360_30fps.mp4',
                      'data/time_lapse_video/biology/2038351-sd_640_360_30fps.mp4',
                      'data/time_lapse_video/biology/855215-sd_640_360_24fps.mp4',
                      'data/time_lapse_video/time_lapse/3541930-sd_426_240_24fps.mp4',
                      'data/time_lapse_video/simple_color_shift/9255202-sd_426_240_25fps.mp4',
                      'data/time_lapse_video/simple_color_shift/9255065-sd_426_240_25fps.mp4',
                      'data/time_lapse_video/bio_mimic/10881636-sd_640_360_25fps.mp4',
                      'data/time_lapse_video/bio_mimic/10881635-sd_640_360_25fps.mp4',
                      'data/time_lapse_video/bio_mimic/10881640-sd_640_360_25fps.mp4',
                  ]

    # Determine if the current video needs trimming
    trim = video_path in videos_to_trim

    # Adjust start and end indices if trimming is needed
    start_index = 10 if trim else 0
    end_index = total_frames - 10 if trim else total_frames

    for frame_count in range(total_frames):
        success, frame = video.read()
        if not success:
            break

        # Skip processing for trimmed frames
        if frame_count < start_index or frame_count >= end_index:
            continue
```

Supplemental Figures

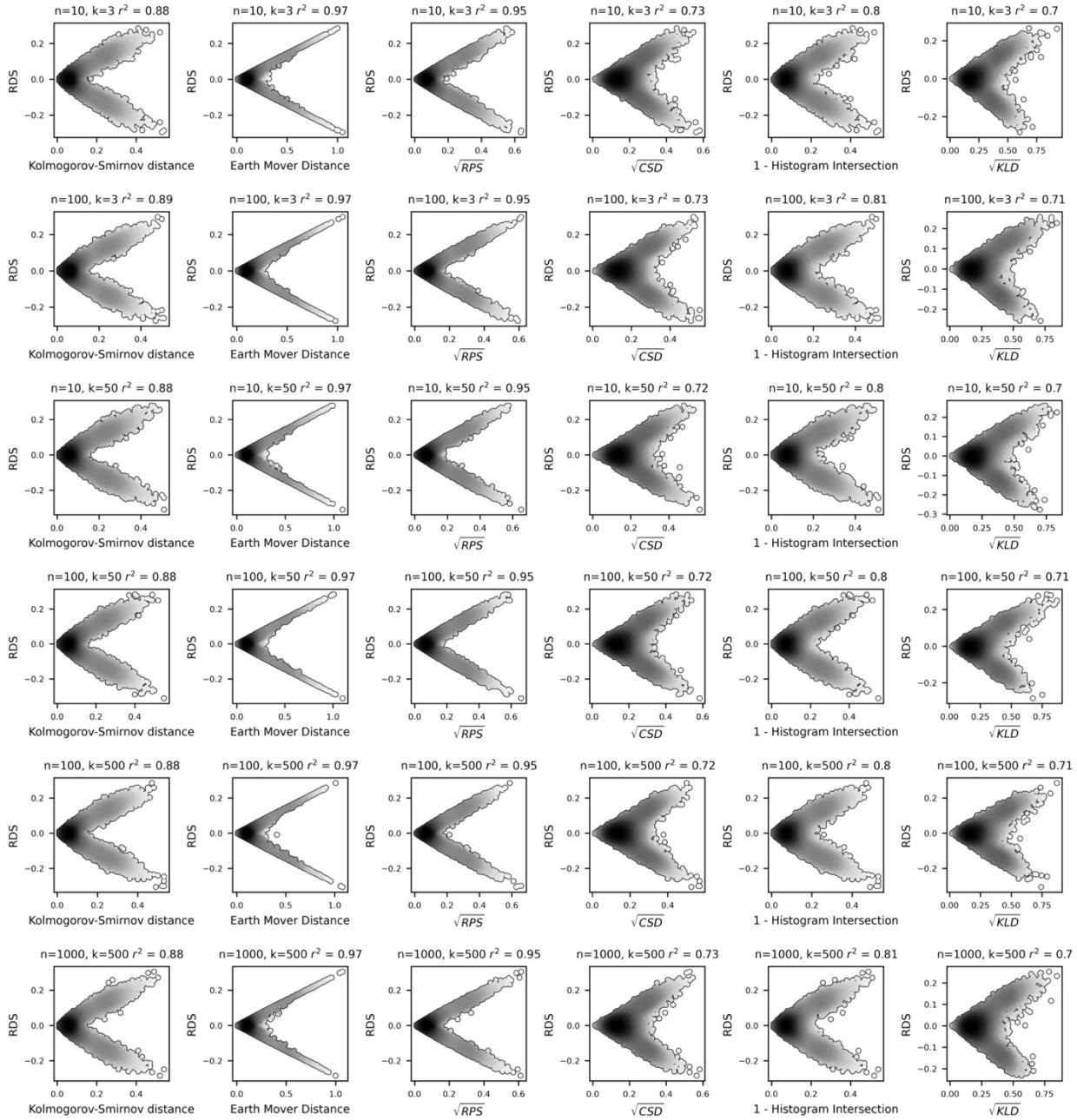
Supplemental figure 1. Results based on Poisson-distributed random samples, wherein each was represented as a histogram of *n* observations distribution among *k* bins.



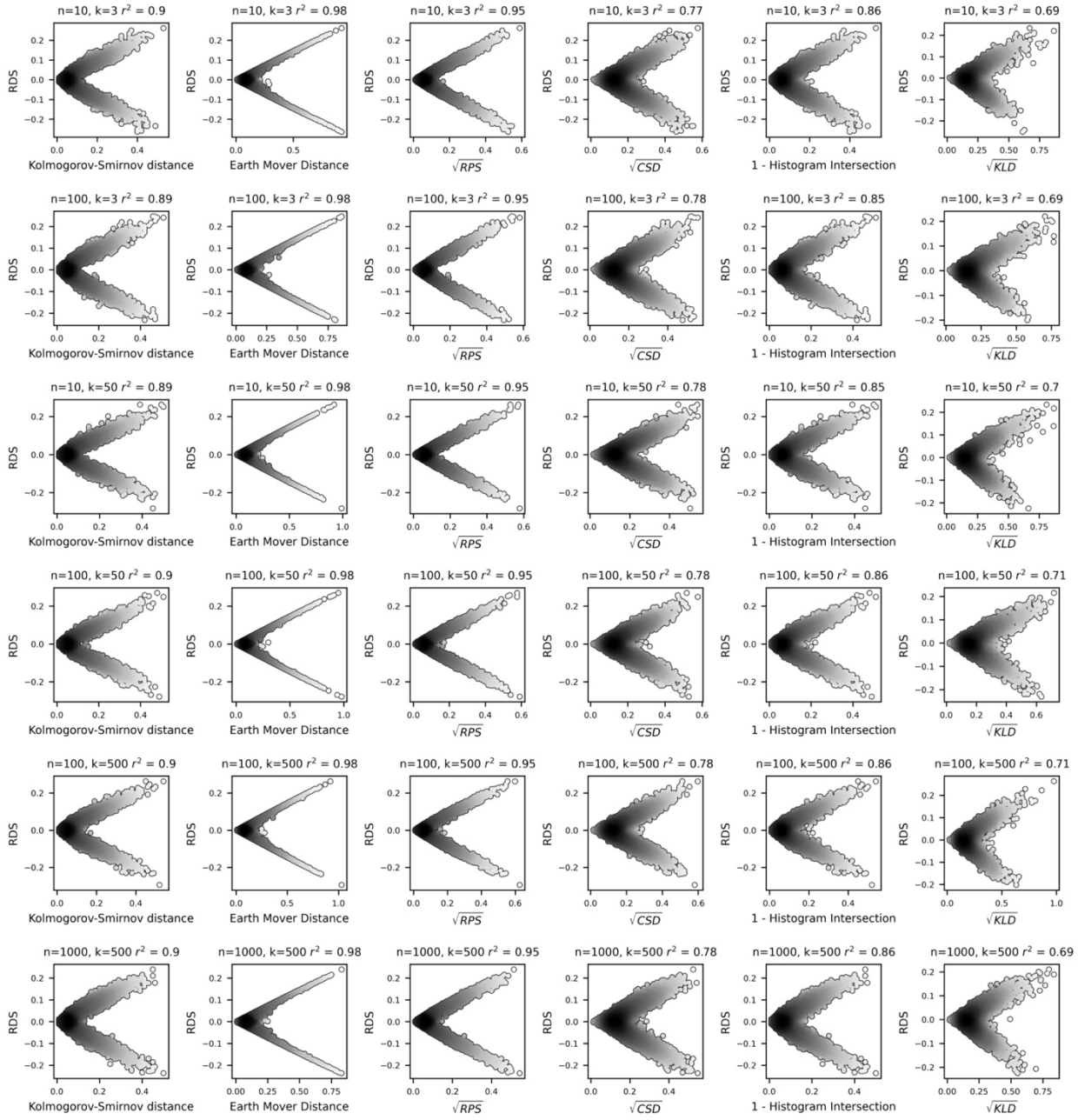
Supplemental figure 2. Results based on Gaussian-distributed random samples, wherein each was represented as a histogram of *n* observations distribution among *k* bins.



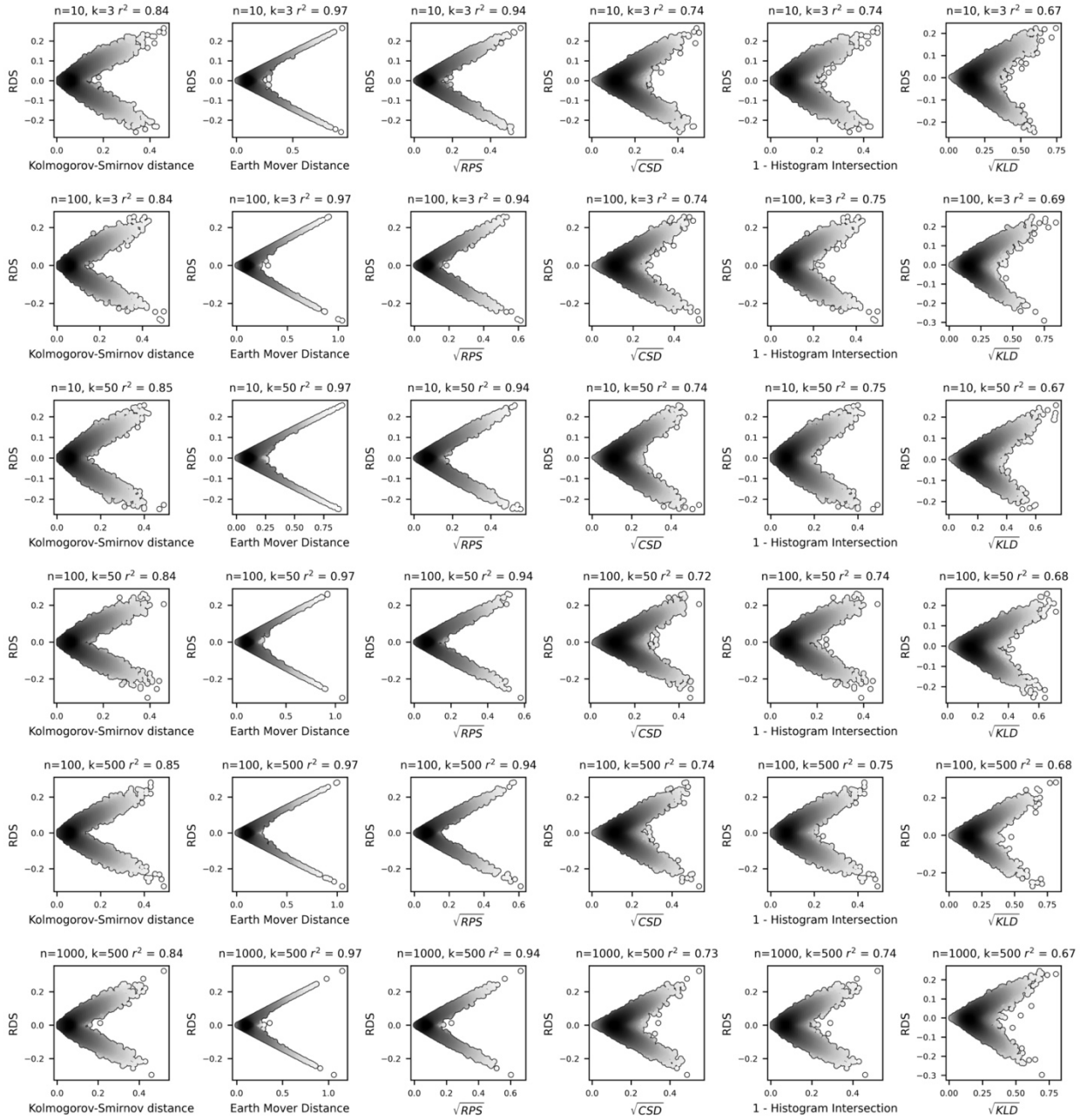
Supplemental figure 3. Results based on Negative binomial-distributed random samples, wherein each was represented as a histogram of *n* observations distribution among *k* bins.



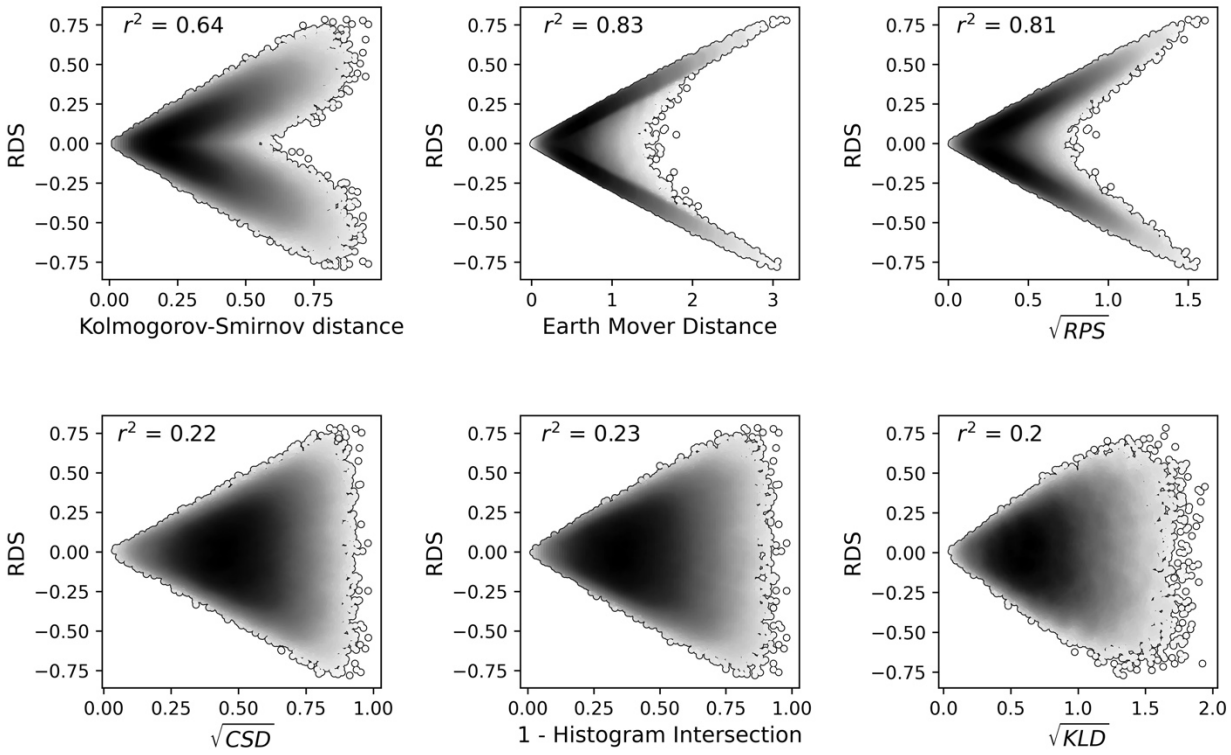
Supplemental figure 4. Results based on Lognormal-distributed random samples, wherein each was represented as a histogram of *n* observations distribution among *k* bins.



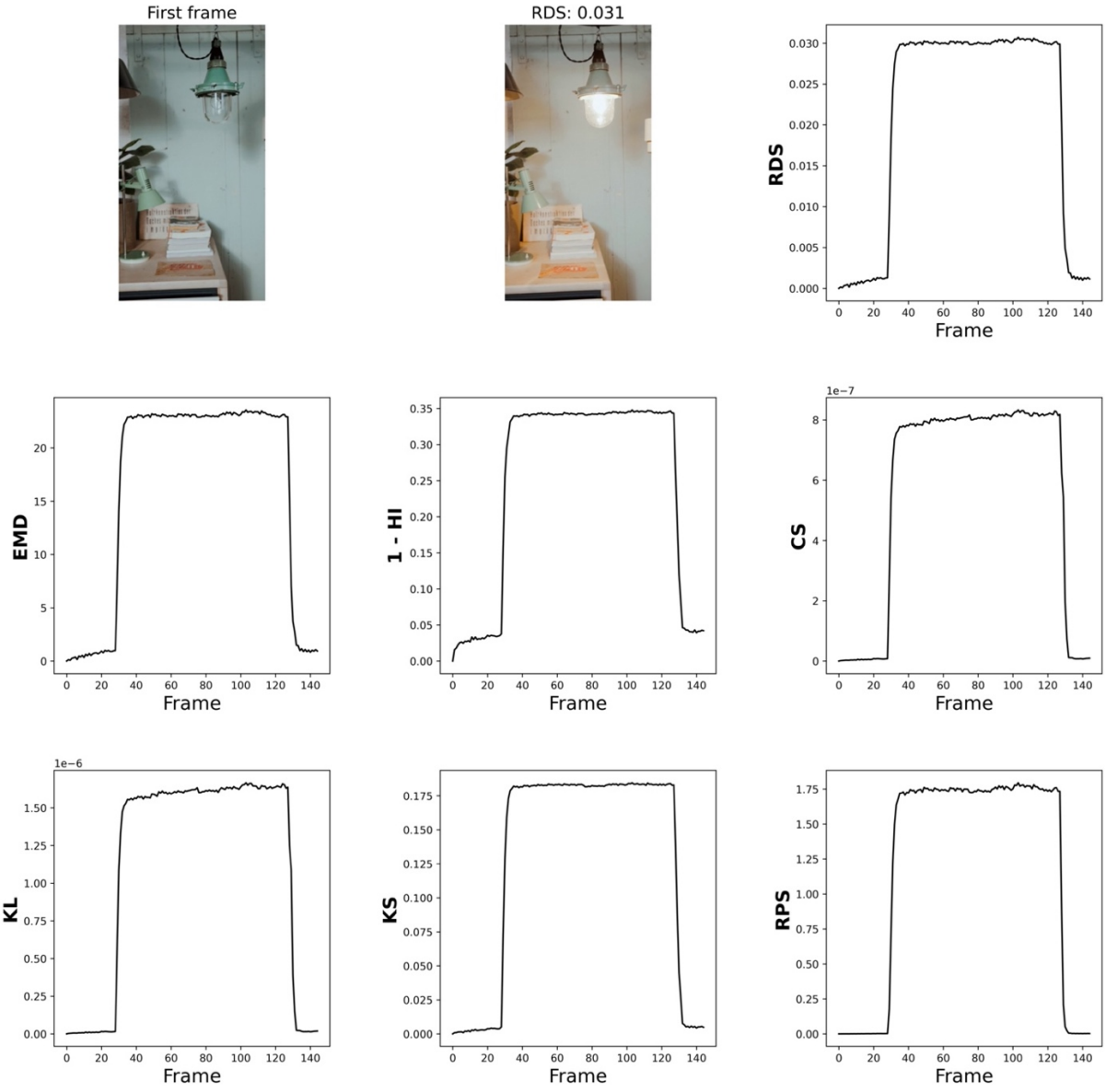
Supplemental figure 5. Results based on Weibull-distributed random samples, wherein each was represented as a histogram of *n* observations distribution among *k* bins.



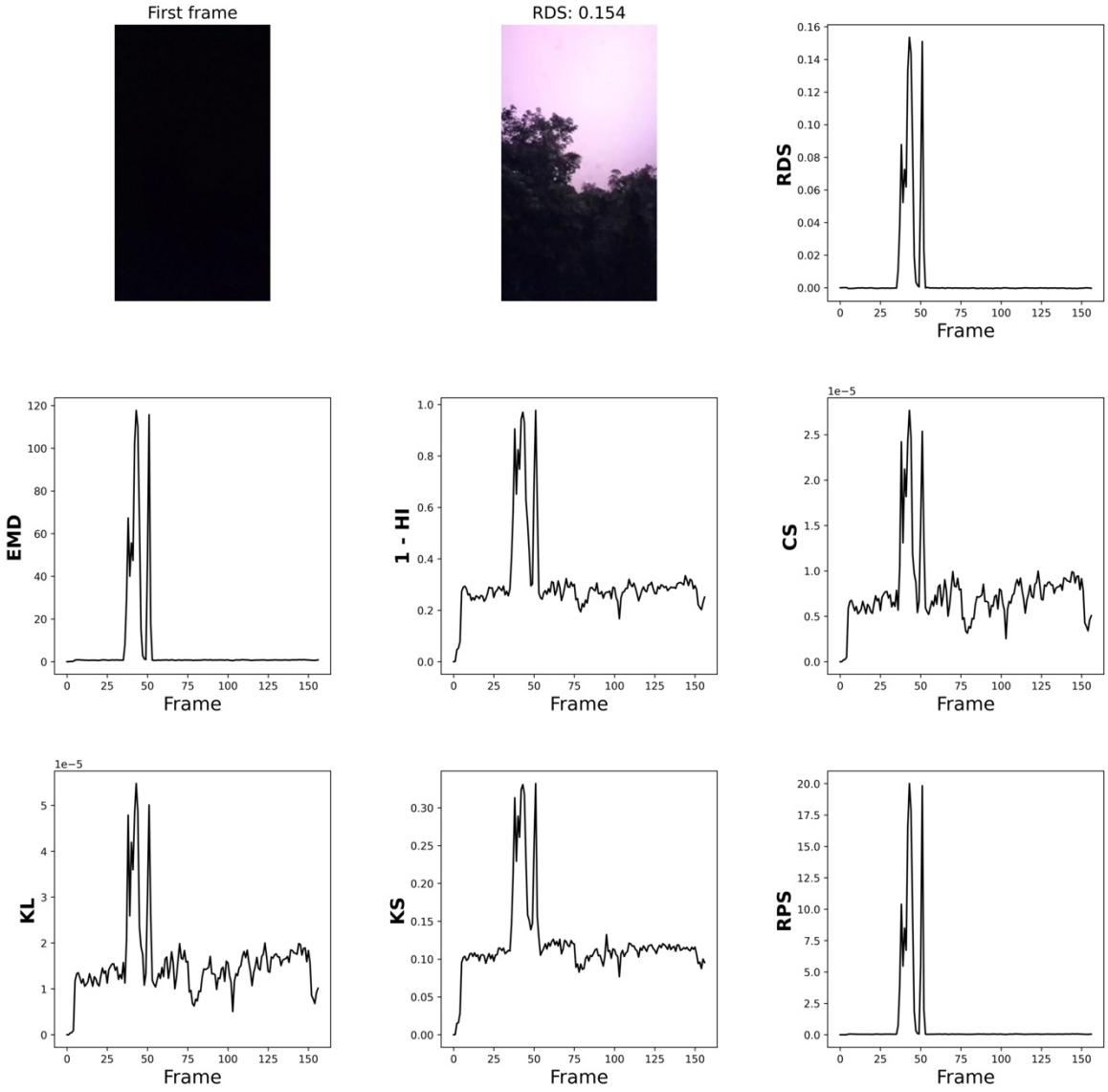
Supplemental figure 6. Results based on randomly drawn pairs of distributions from the feasible set of 4,598,126 distributions satisfying $n = 100$ and $k = 5$. Shown are relationships of Relative Distributional Shift (RDS) to Kolmogorov-Smirnov distance, Earth Mover distance, Ranked Probability Score (RPS), Chi-square distance (CSD), histogram non-intersection, and Kullback-Leibler divergence (KLD). Coefficients of determination (r^2) were calculated using |RDS| and simple linear regression.



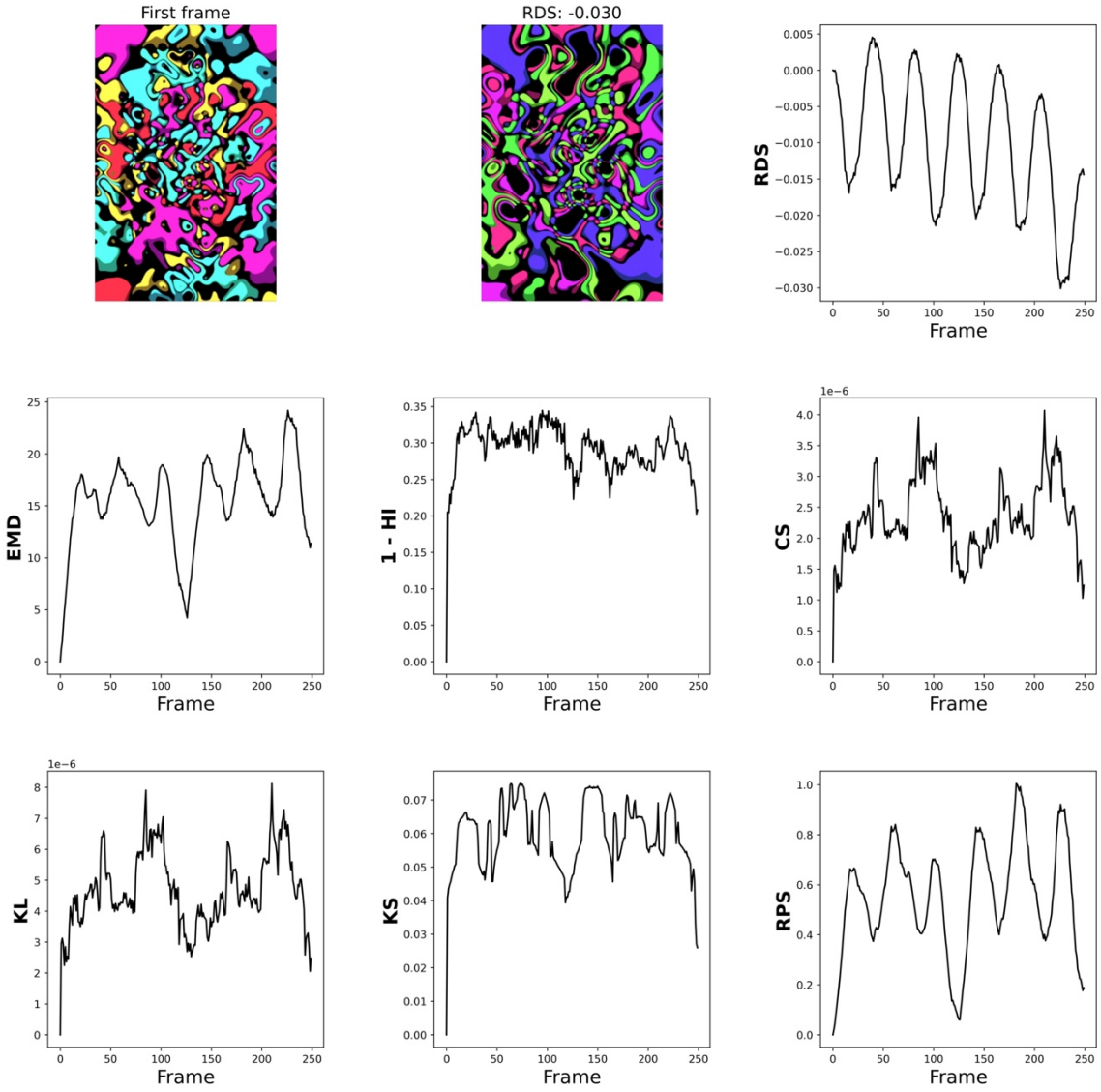
Supplemental figure 7. Video analysis. Detection of a light bulb that turns on and then off. Top right: The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All other plots present comparisons of the first frame to subsequent frames using RDS, Earth Mover's Distance (EMD), histogram non-intersection (1 – HI), Chi-Square Distance (CS), Kullback-Leibler Divergence (KL), Kolmogorov-Smirnov Distance (KS), and Ranked Probability Score (RPS). All measures produce similar curves.



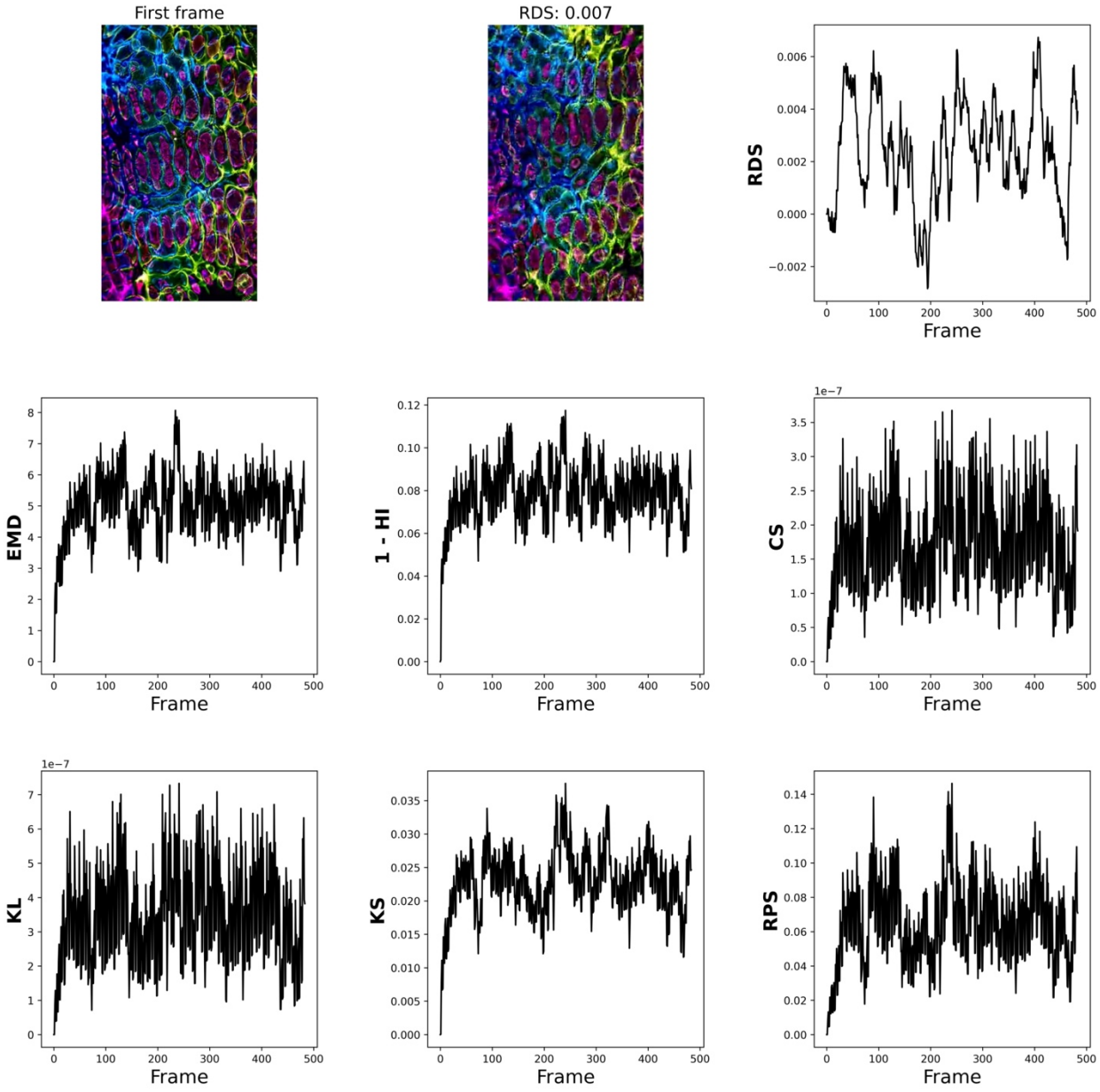
Supplemental figure 8. Video analysis. Detection of a short burst of lightening strikes against a black background. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All other plots present comparisons of the first frame to subsequent frames using RDS, Earth Mover's Distance (EMD), histogram non-intersection (1 – HI), Chi-Square Distance (CS), Kullback-Leibler Divergence (KL), Kolmogorov-Smirnov Distance (KS), and Ranked Probability Score (RPS). Curves for 1 – HI, CS, KL, and KS contain considerable noise.



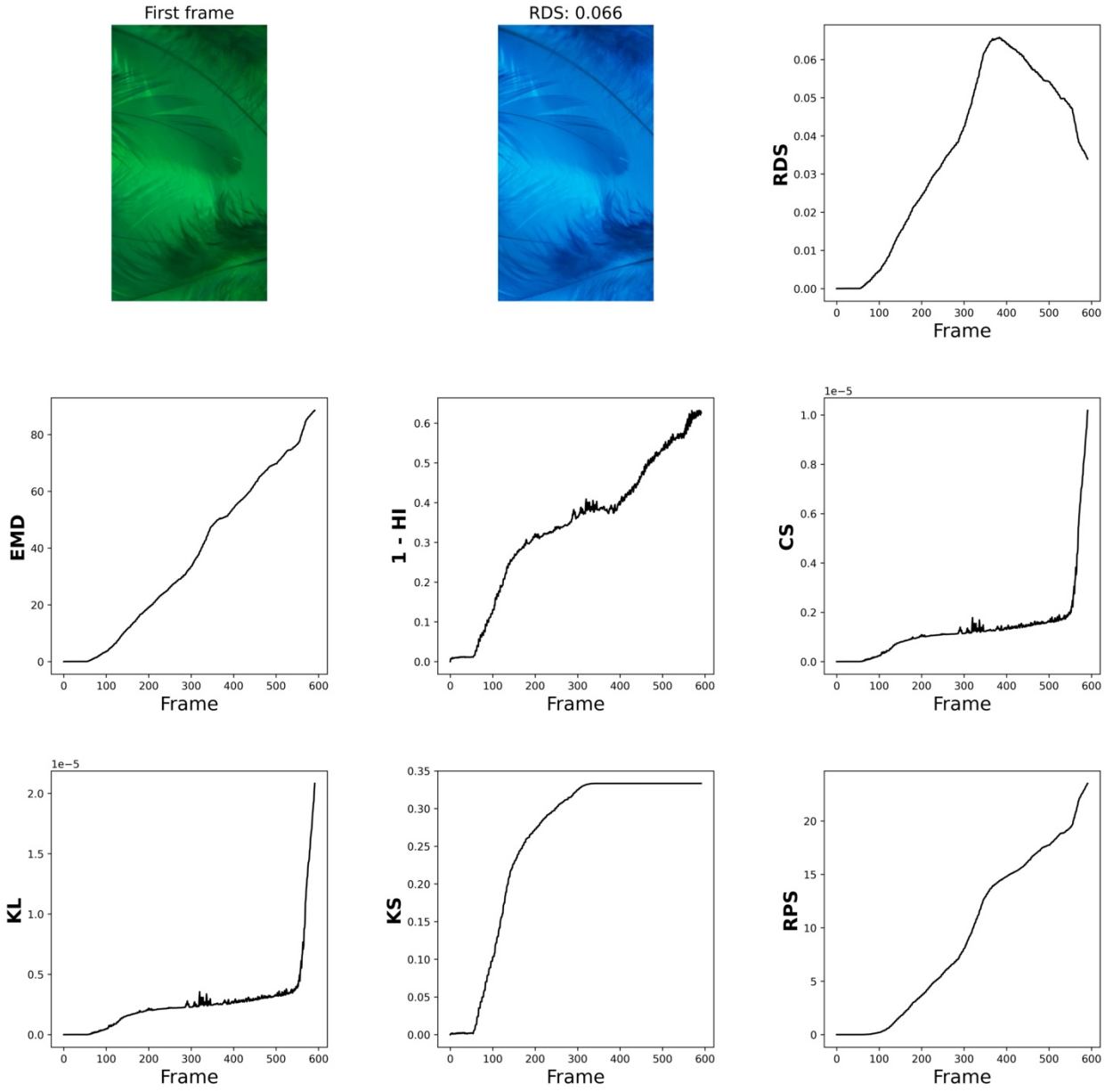
Supplemental figure 9. Video analysis. Detection of complex color patterns among amorphously changing shapes visually shifting from hues of red, to orange, yellow, green, blue, purple, magenta, and back to red in synchronous and asynchronous fashion. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All other plots present comparisons of the first frame to subsequent frames. RDS captures a highly sinusoidal pattern that negatively trends towards red-shift.



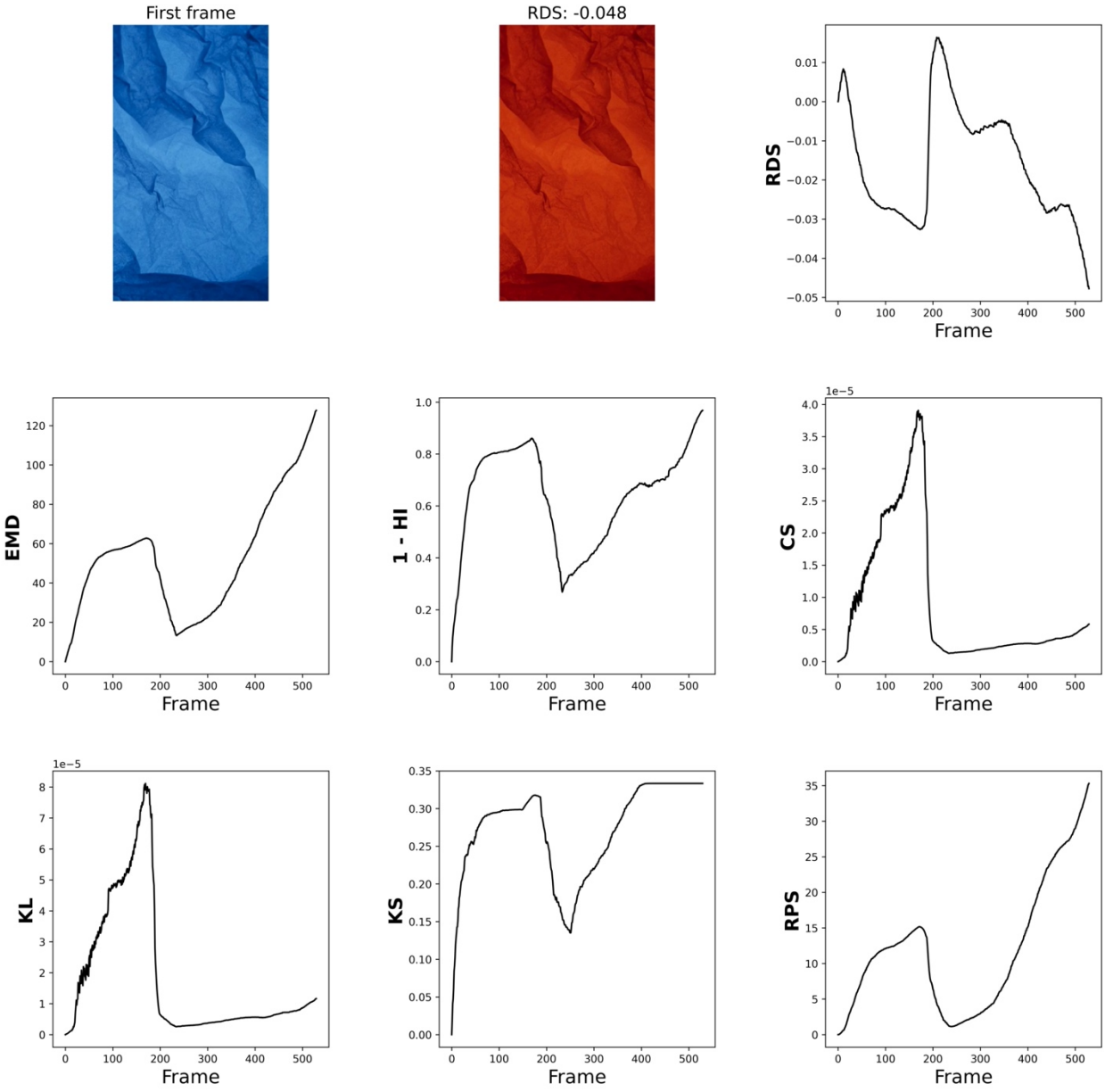
Supplemental figure 10. Detection of complex color changes within a digital art video resembling an animation of fluorescent cell imagining. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All other plots present comparisons of the first frame to subsequent frames. RDS captures an irregular pattern of considerably less noise than curves produced by other measures.



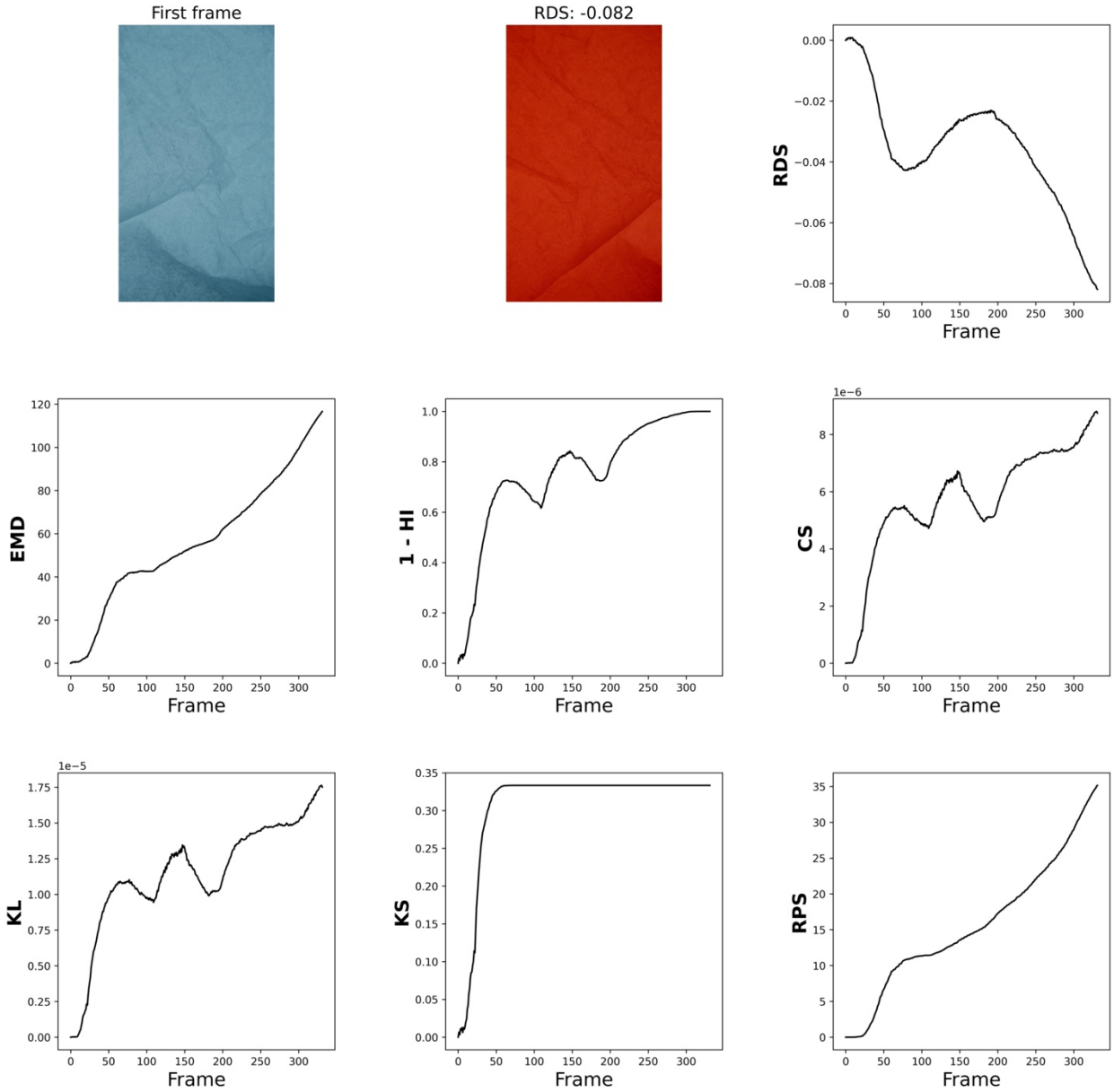
Supplemental figure 11. Detection of red-shift and blue-shift from a video wherein a static image transitioned through shades of color. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All other plots present comparisons of the first frame to subsequent frames. RDS was the only measure that registered an internal mode, i.e., a gradual blue-shift away from the reference image, followed by a gradual red-shift towards the reference image.



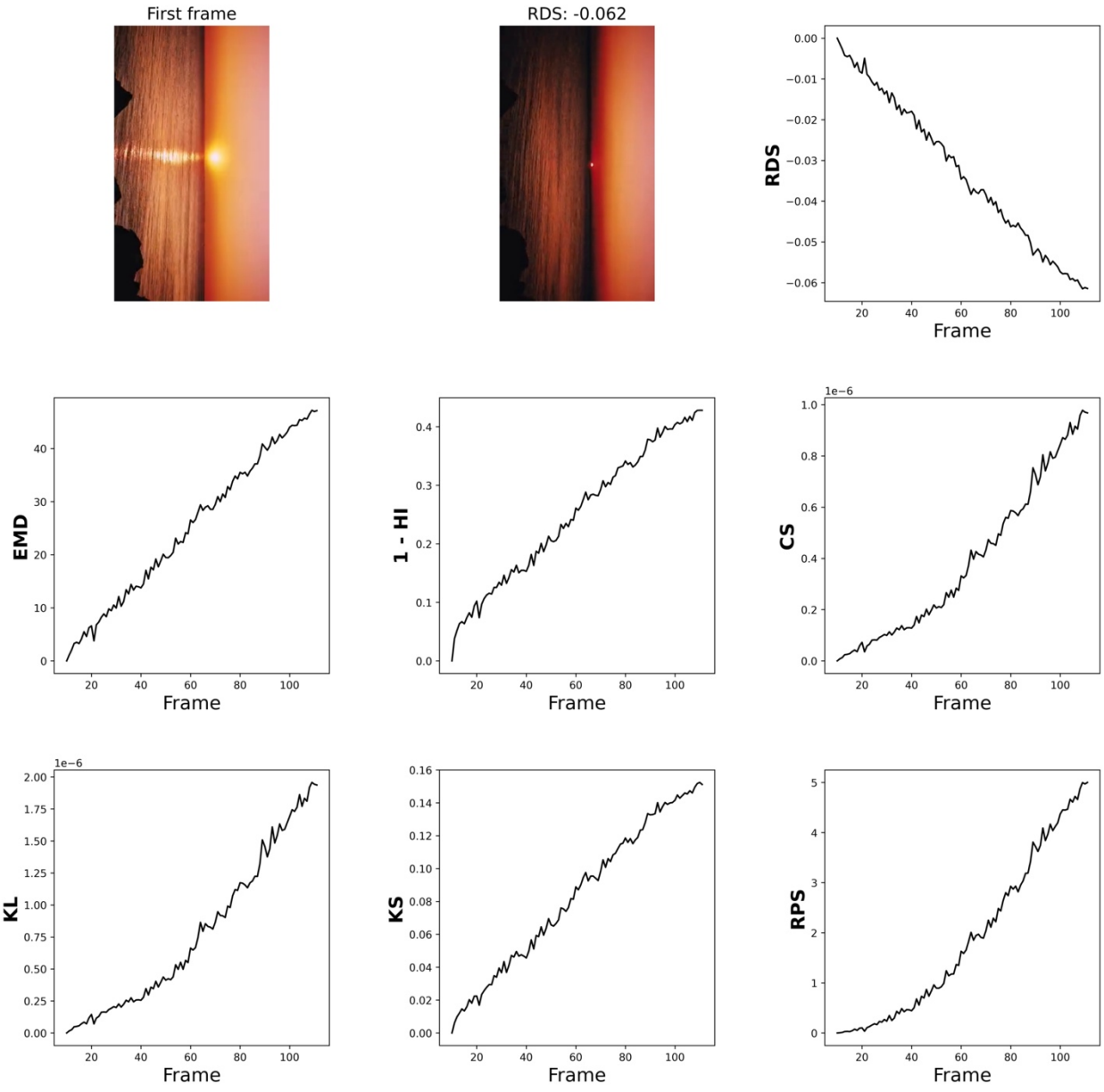
Supplemental figure 12. Detection of red-shift and blue-shift from a video wherein a textured image transitions through shades of color. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All other plots present comparisons of the first frame to subsequent frames. RDS is the only measure that can distinguish red-shifts from blue-shifts.



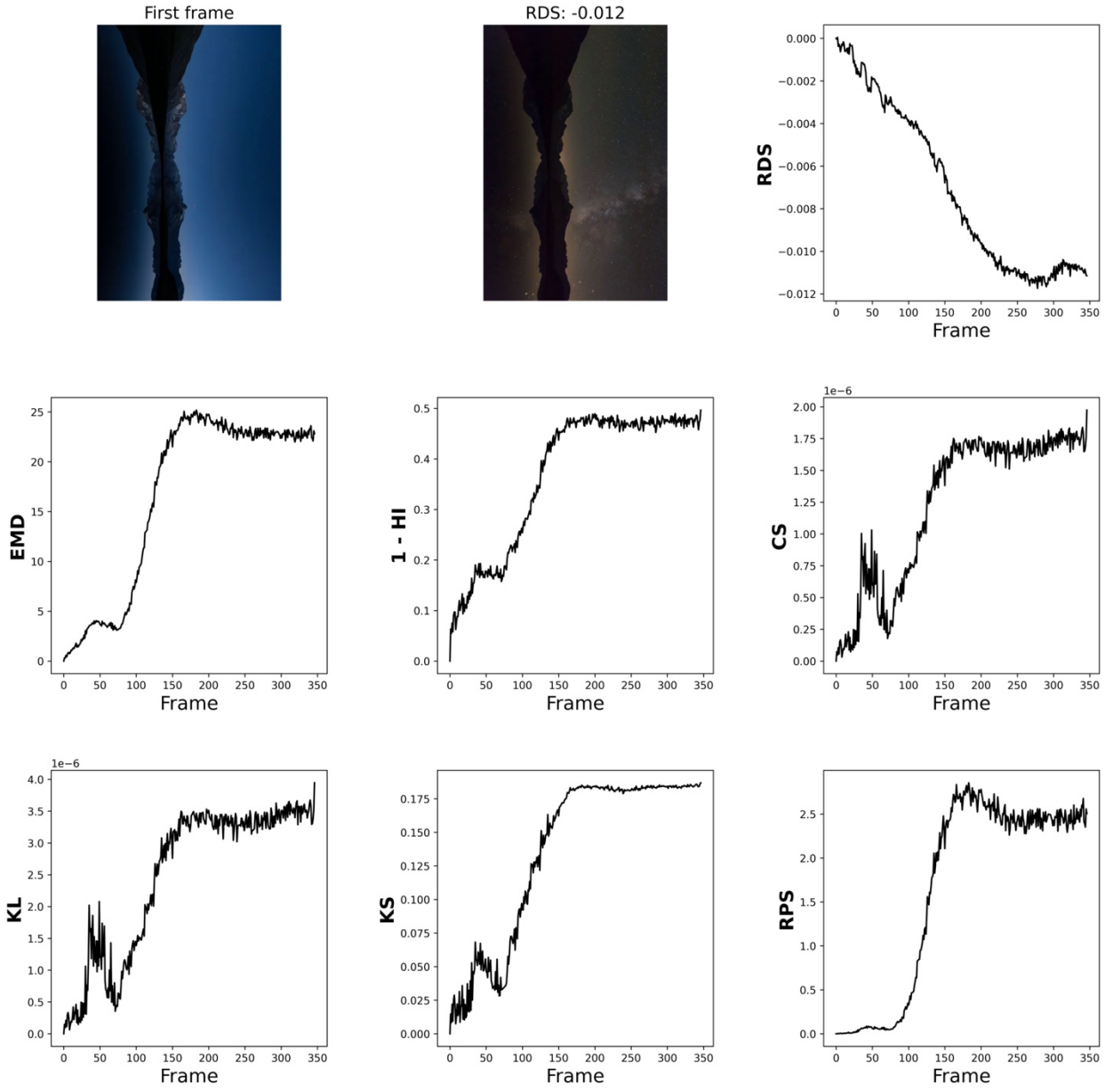
Supplemental figure 13. Detection of red-shift and blue-shift from a video wherein a textured image is panned as it transitions through shades of color. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All other plots present comparisons of the first frame to subsequent frames. RDS is the only measure that can distinguish red-shifts from blue-shifts.



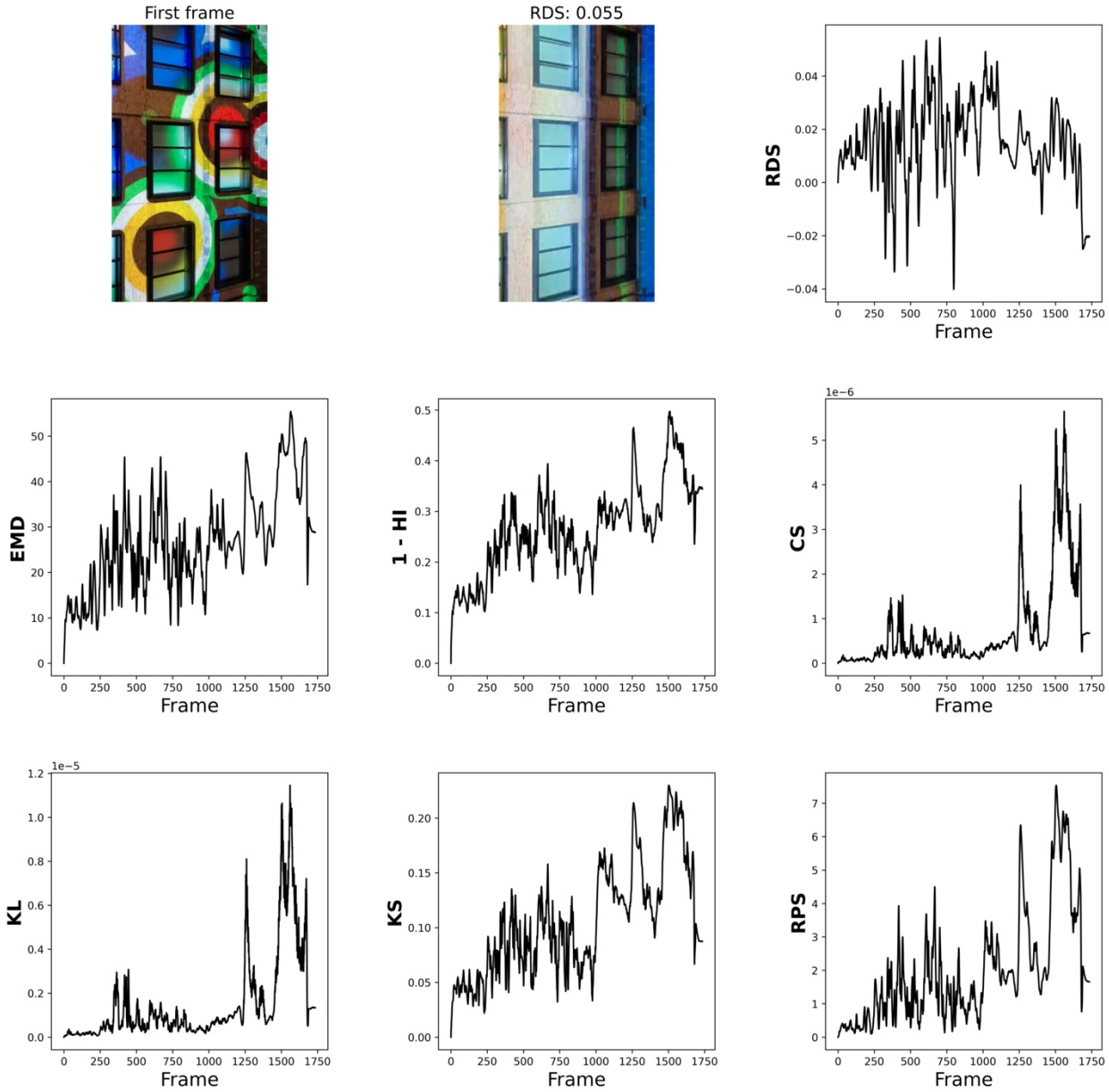
Supplemental figure 14. Time-lapse video of a sunset. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All measures signal a monotonic increase in difference from the reference image. RDS indicates that the change is a general red-shift.



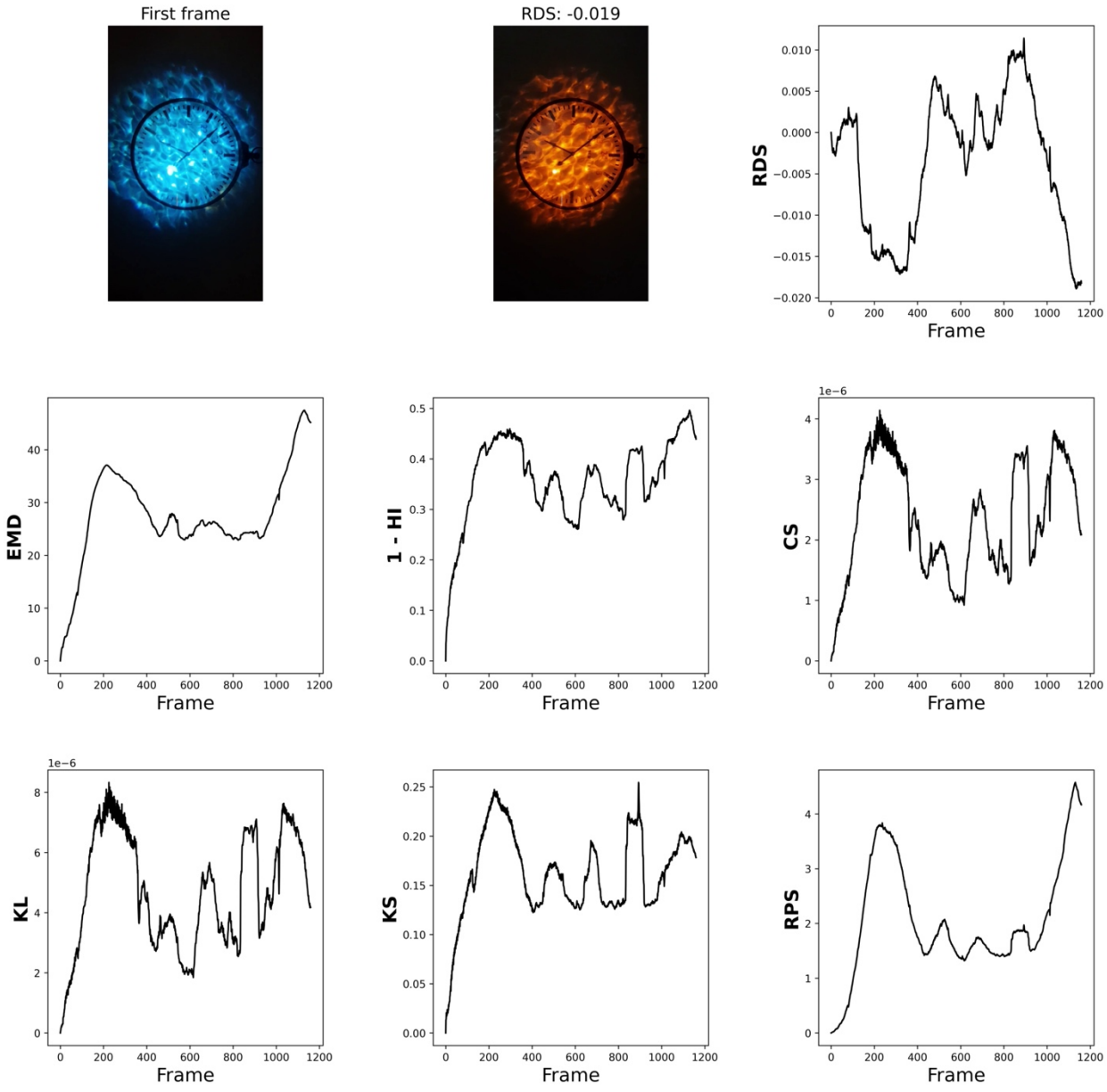
Supplemental figure 15. Time-lapse video of early-to-late evening. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. RDS signals a largely monotonic red-shift which is then disrupted by the shifting appearance of the Milky Way. Other measures signal an asymptote of difference from the reference frame. EMD, 1 – HI, CS, KL, and KS signal an abrupt light event that is not visually apparent in the corresponding video.



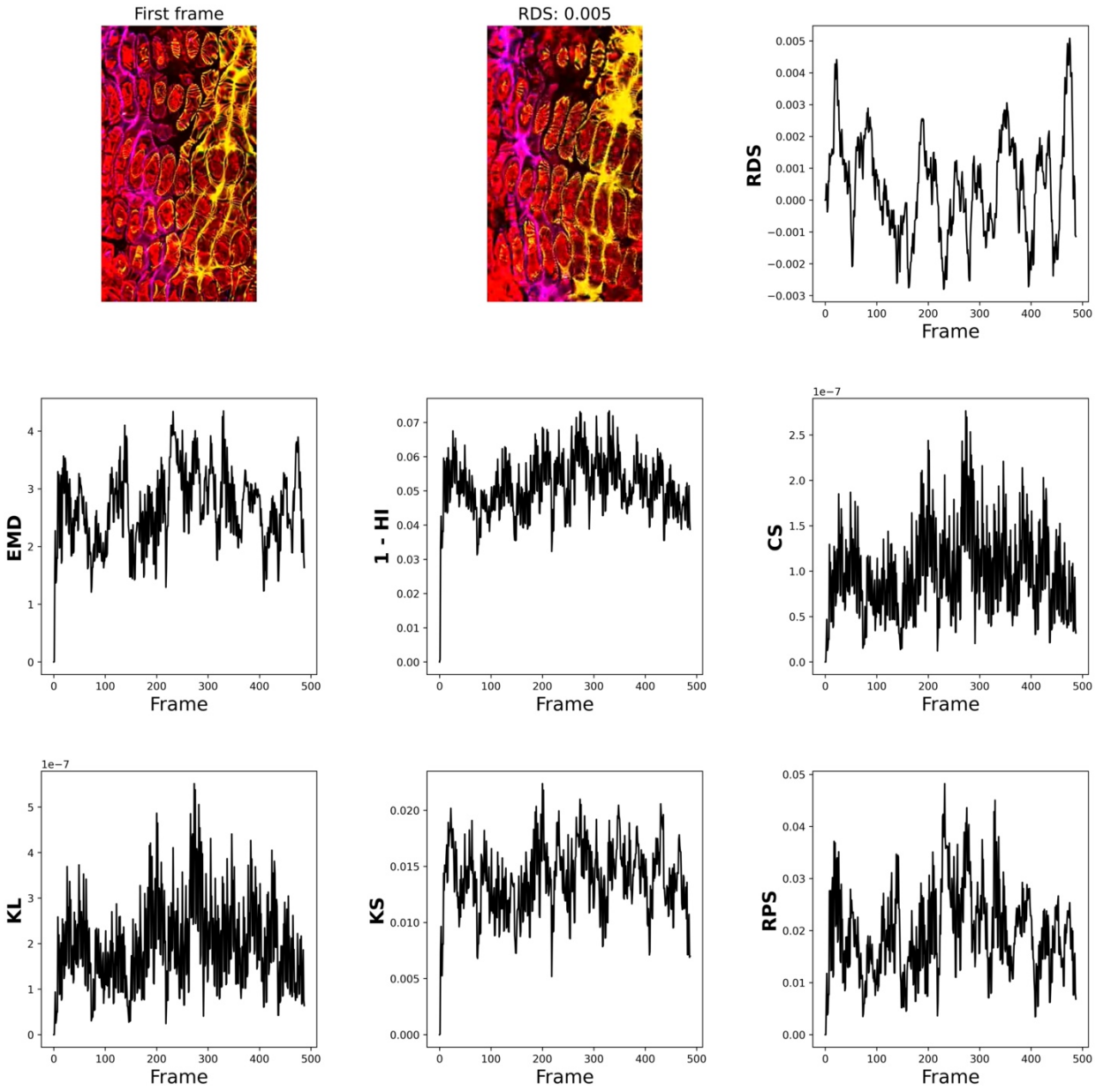
Supplemental figure 16. Detection of red-shift and blue-shift from a video wherein abstract art is projected against a building. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. RDS registers noisy red-shifts and blue-shifts of generally similar maximum magnitude (from ca. -0.055 to 0.055) followed by a trend towards red-shift. Other measures register a noisy trend towards greater differences (EMD, 1 – HI, KS) from the first frame, or no trend at all.



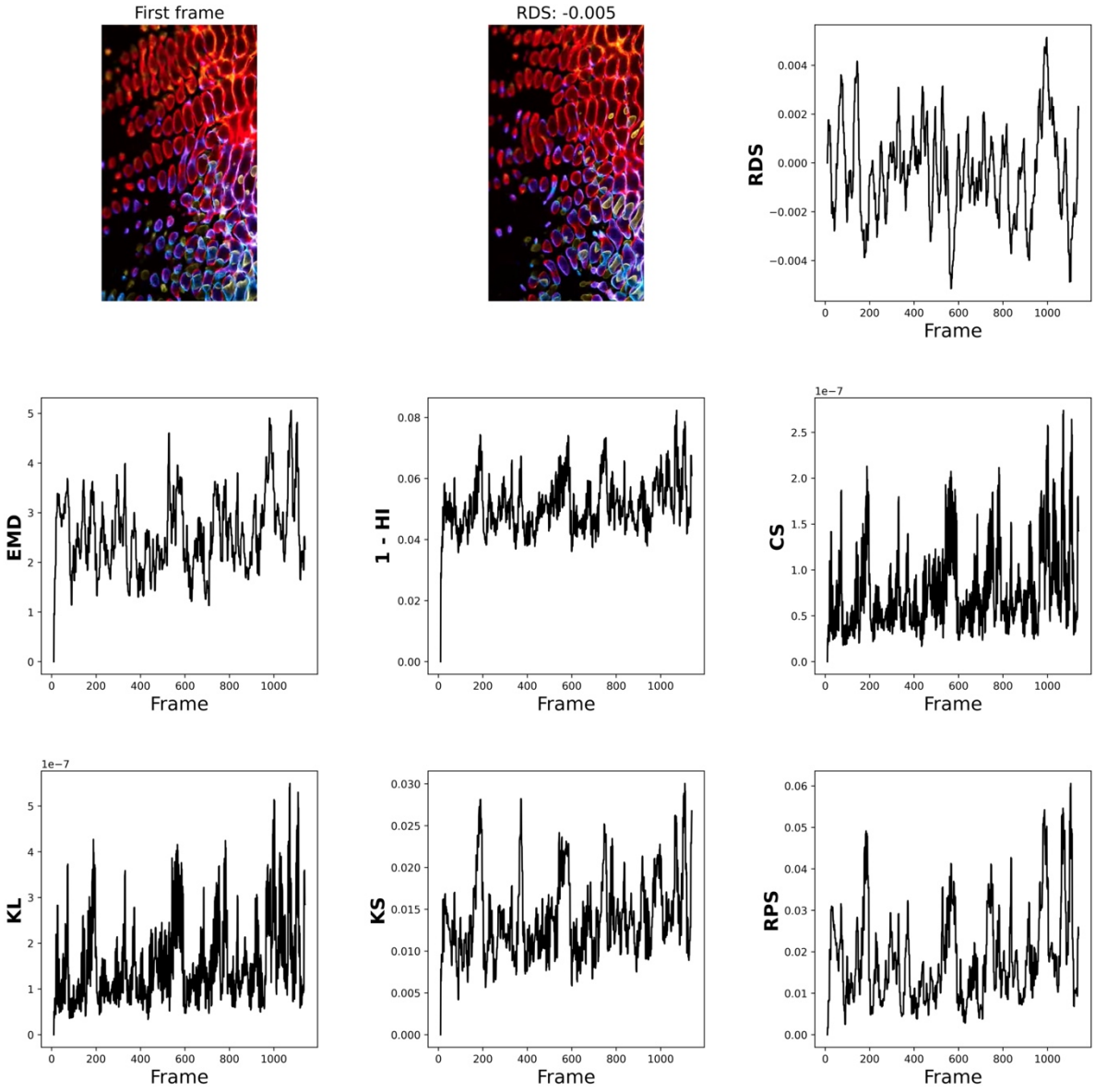
Supplemental figure 17. Detection of red-shift and blue-shift from a video wherein an image of a clock shifts from blue hues to red-orange hues. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. Each measure signals a change but only RDS indicates directional shift.



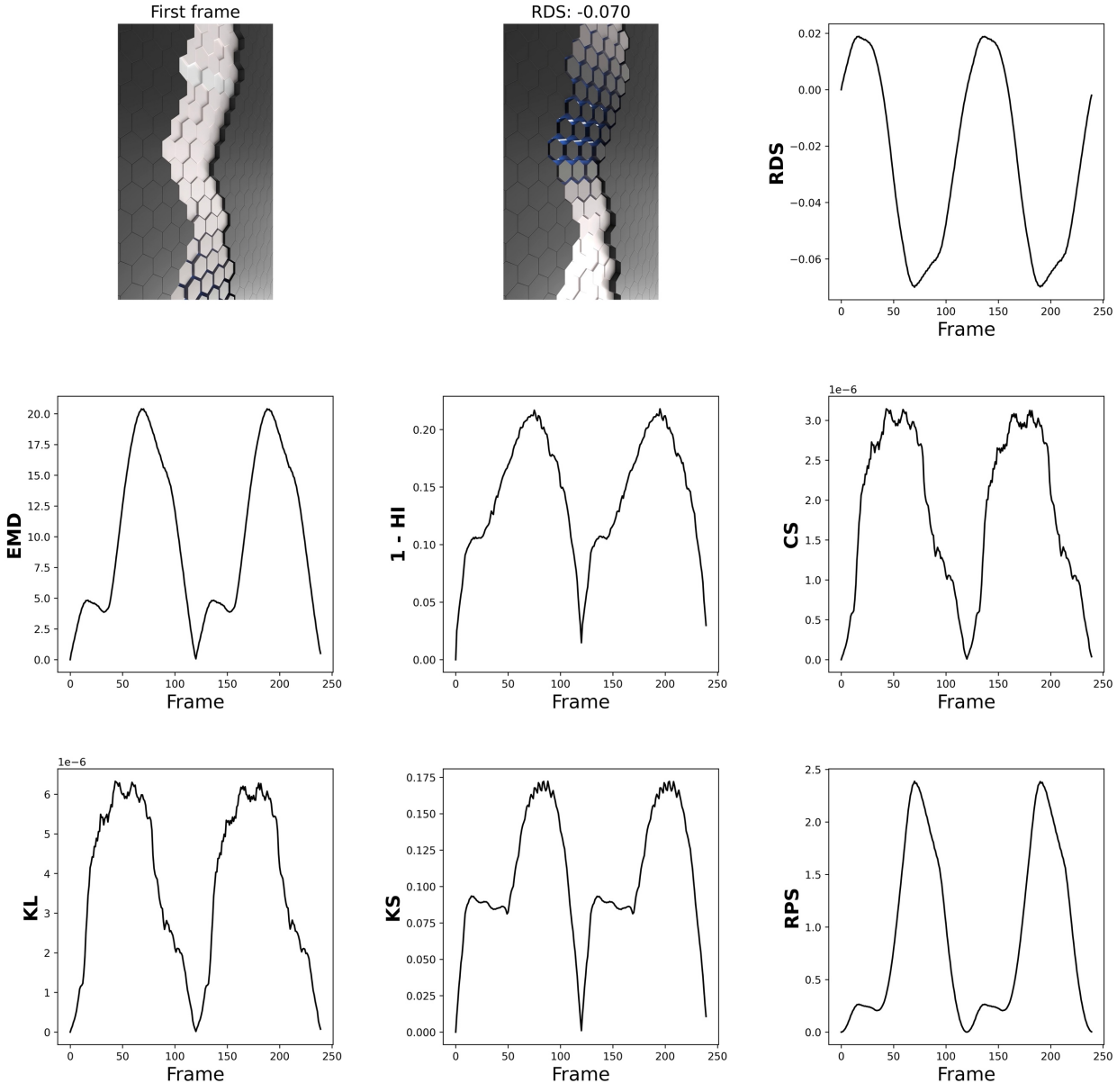
Supplemental figure 18. Detection of complex color changes within a digital art video resembling an animation of fluorescent cell imagining. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. RDS captures an irregular pattern of less noise than curves produced by other measures.



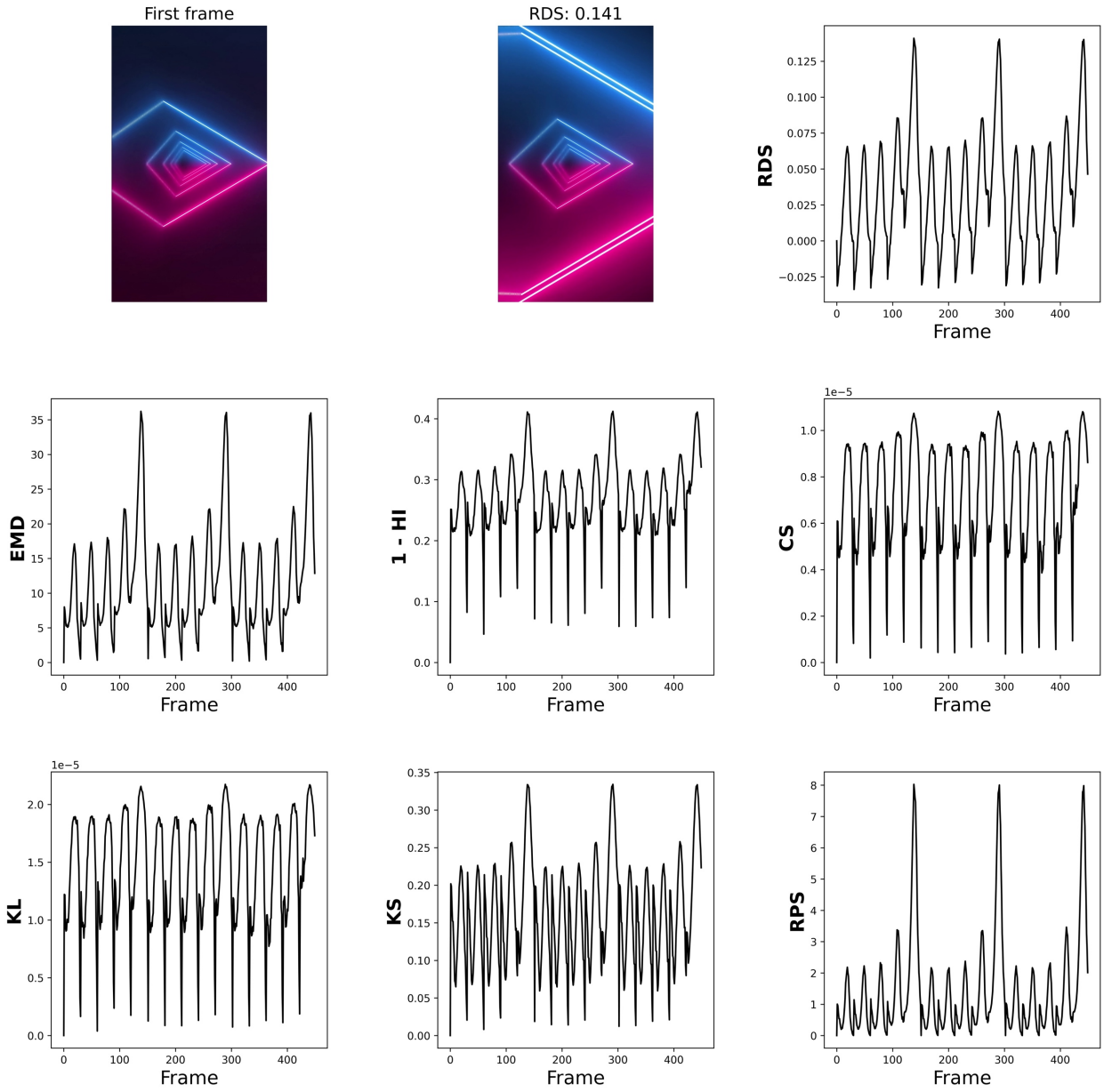
Supplemental figure 19. Detection of complex color changes within a digital art video resembling an animation of fluorescent cell imagining. The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All measures produced erratic signals but that of RDS was less noisy than others and indicated generally similar maximum magnitudes of red-shift and blue-shift (from ca. -0.005 to ca 0.005).



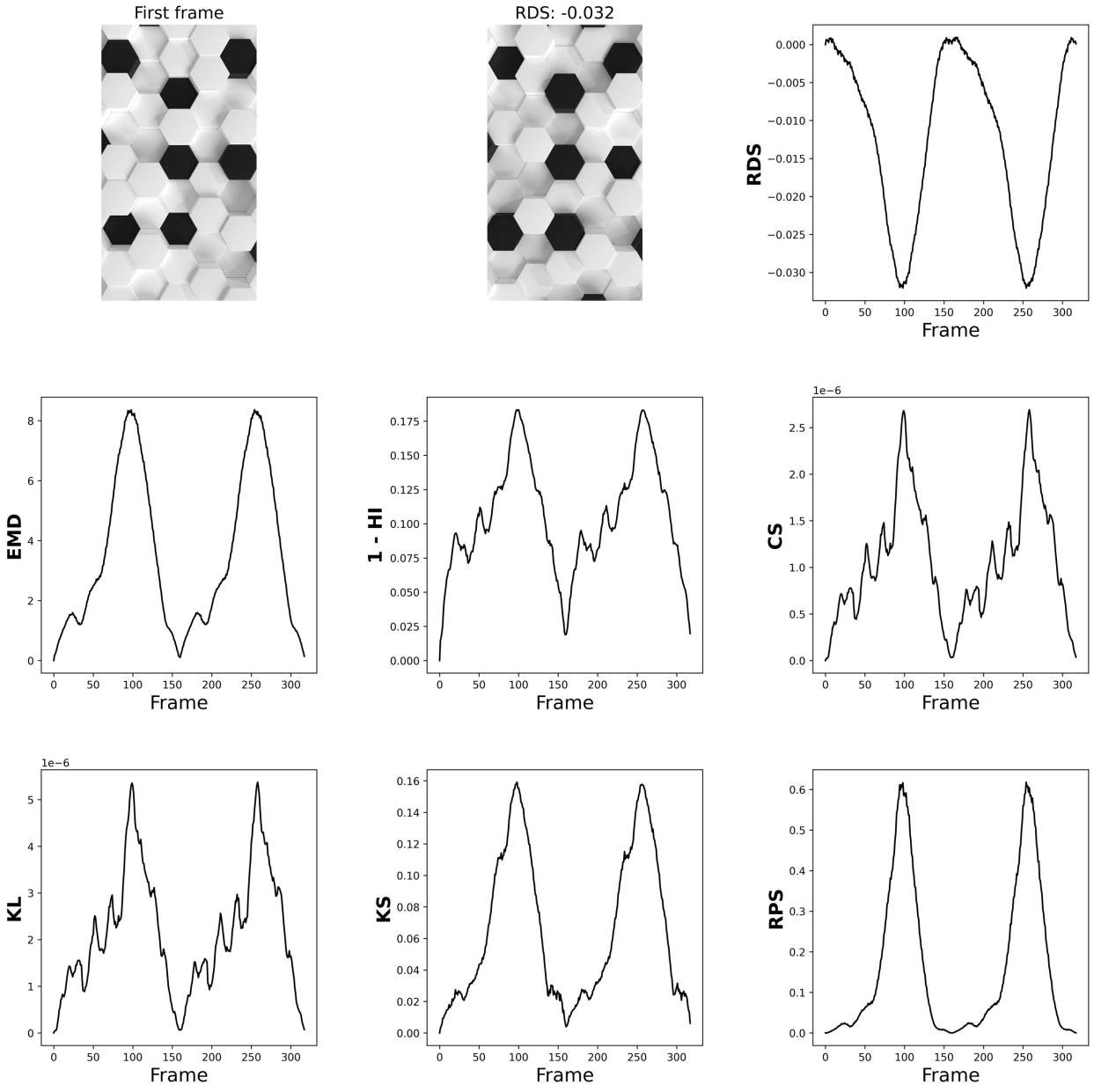
Supplemental figure 20. Detection of signal within a video of moving hexagons. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value, i.e., greatest difference from the first frame as measured via RDS. RDS produces a simpler, more regular signal because it distinguishes positive and negative shift.



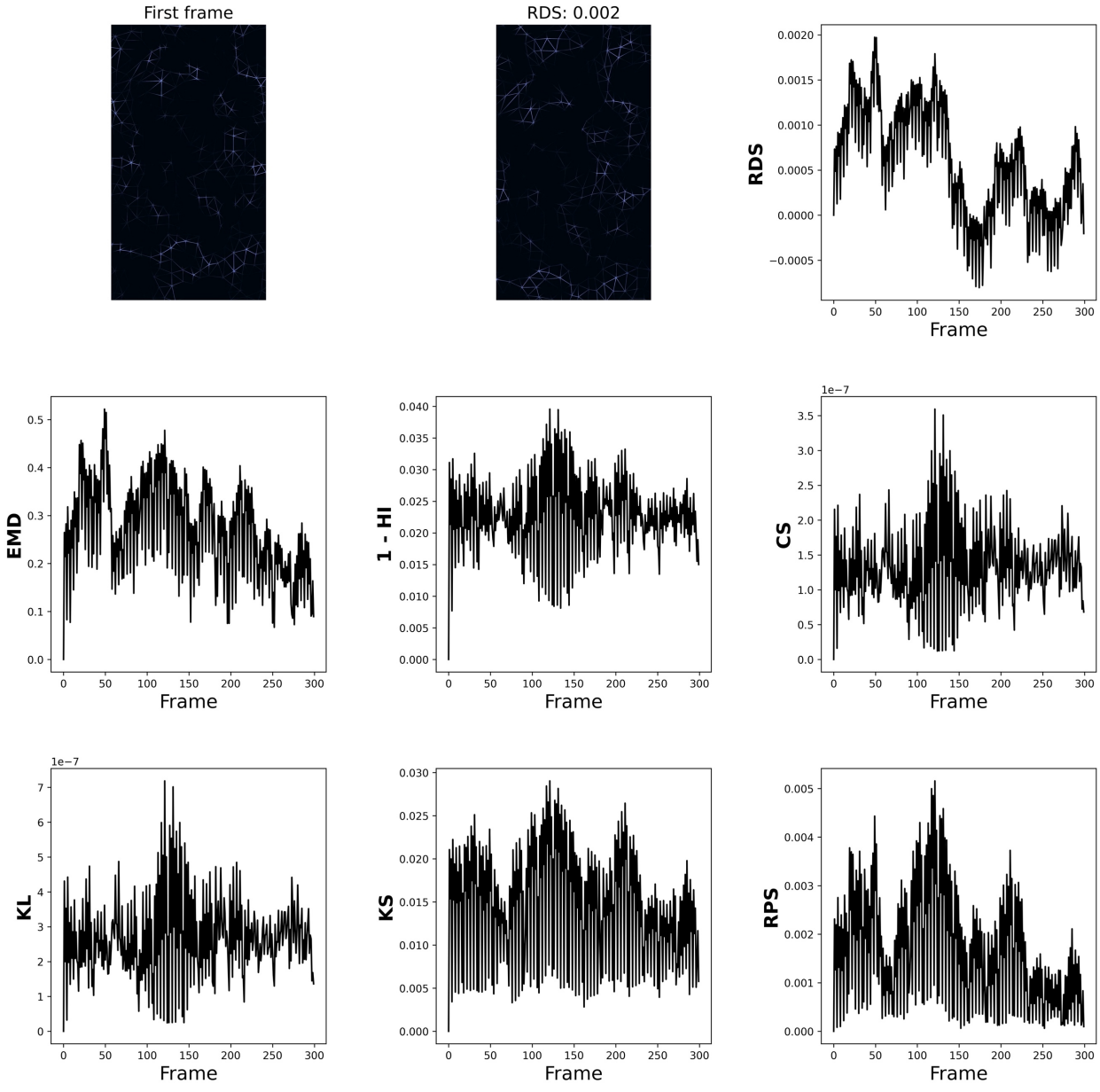
Supplemental figure 21. Detection of signal in a video of contrasting polygons. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS produces a simpler, more regular signal because it distinguishes positive and negative shift.



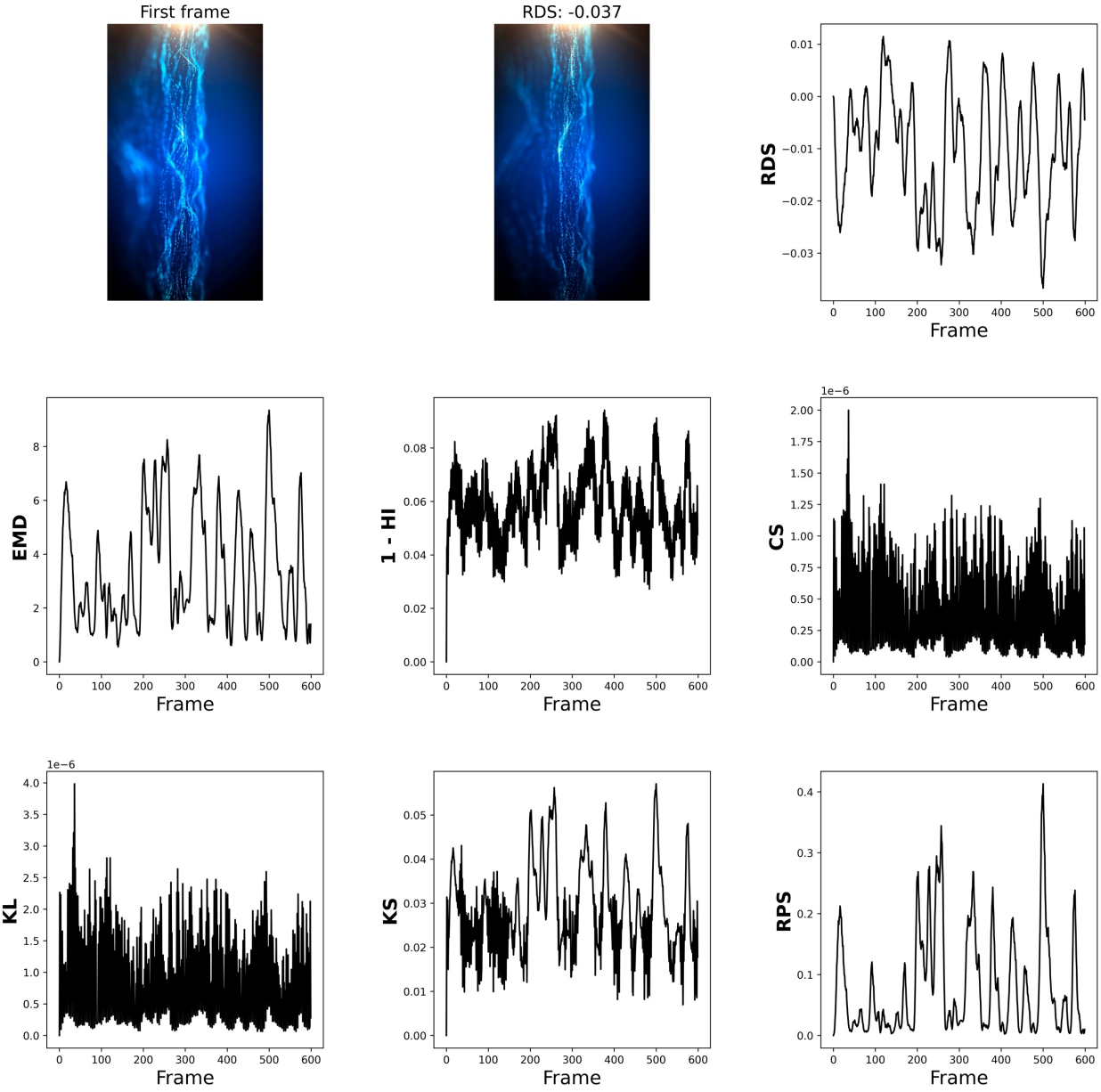
Supplemental figure 22. Detection of signal within a video of moving hexagons. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS produces a simpler, more regular signal because it distinguishes positive and negative shift.



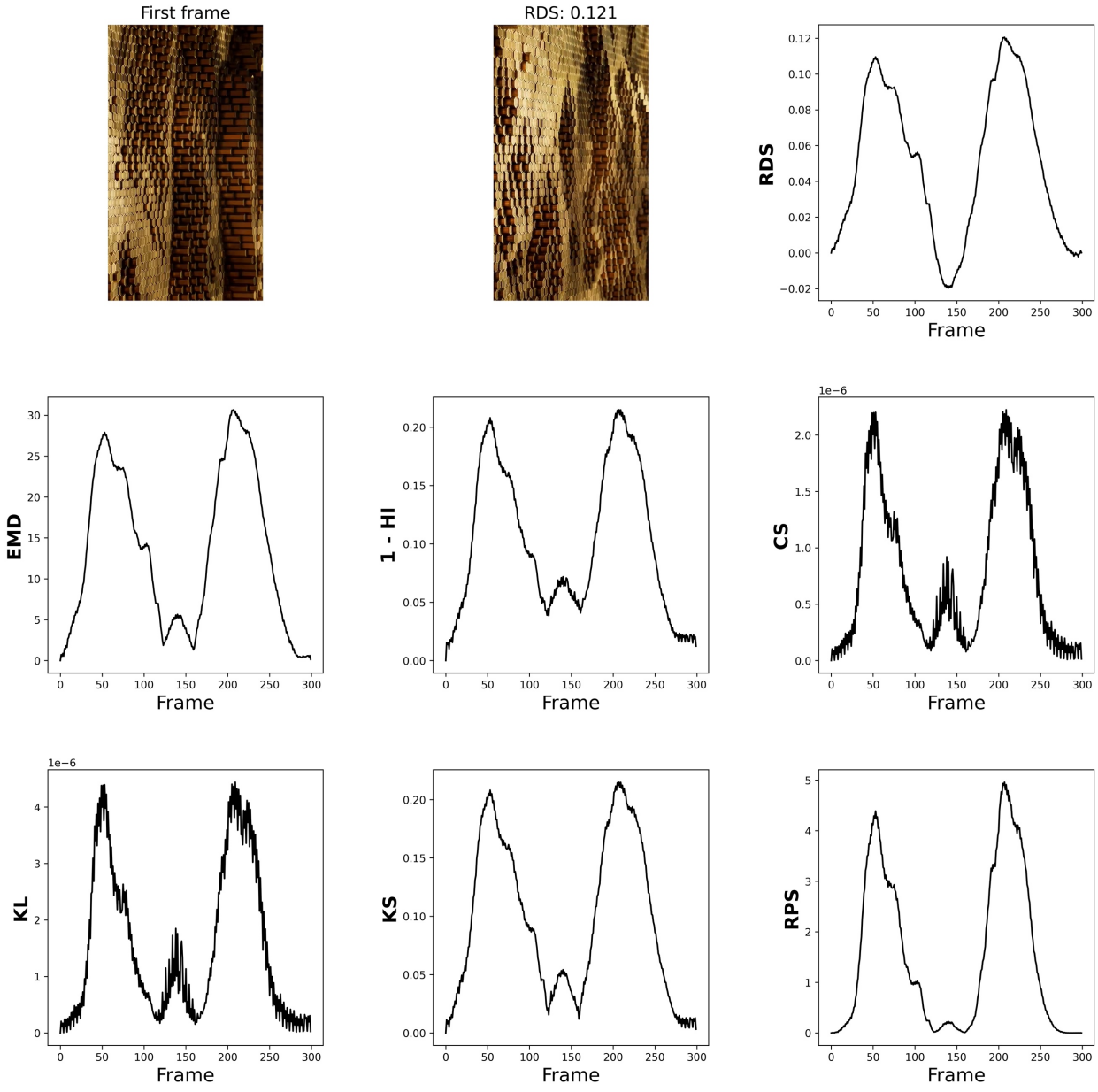
Supplemental figure 23. Detection of signal within a low-contrast video of lines segments and polygons. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS produces a more discernable signal because it distinguishes positive and negative shift.



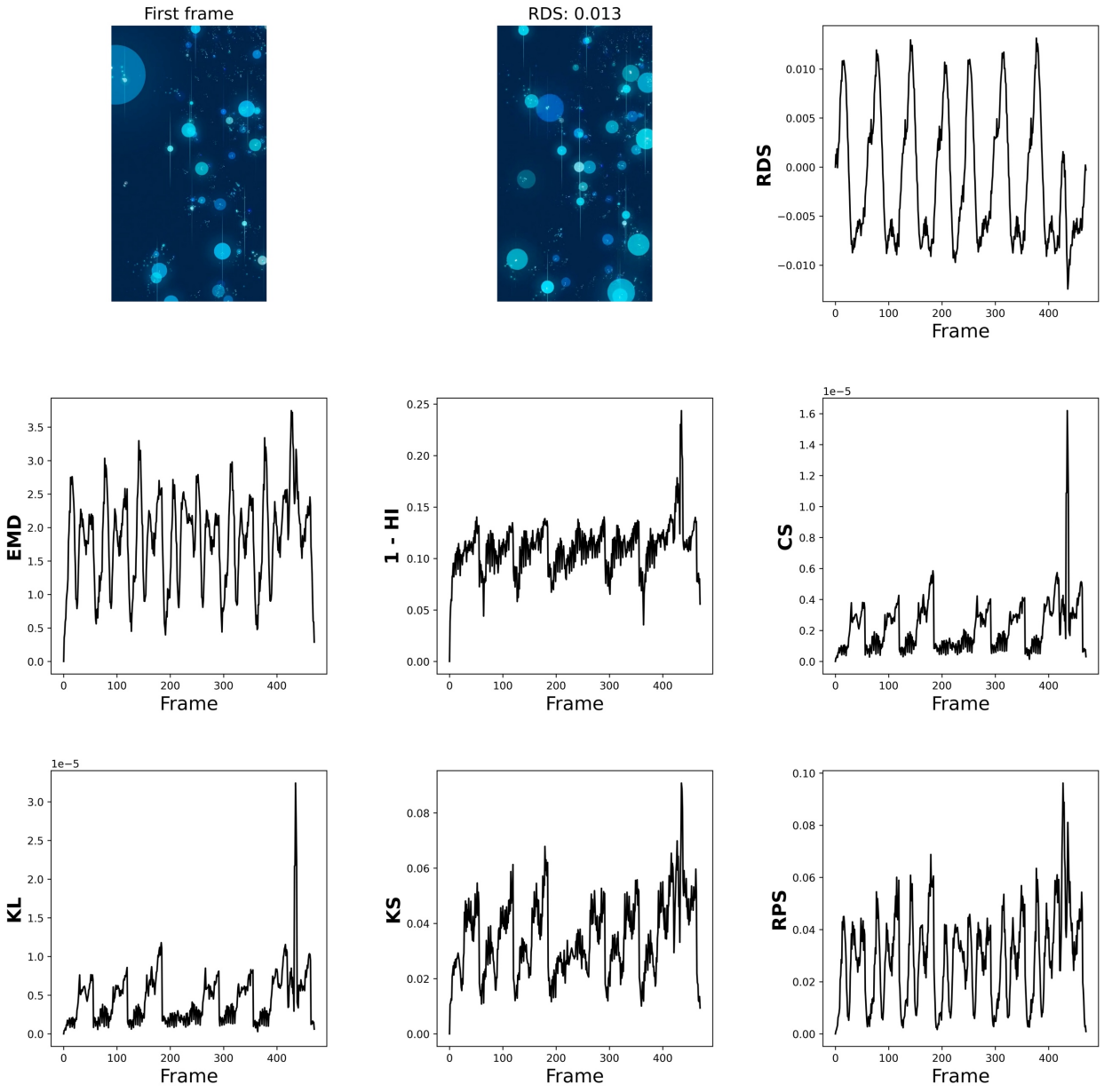
Supplemental figure 24. Detection of signal within a video of illuminated waves. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS produces a less noisy signal than 1 – HI, CS, and KL and distinguishes positive and negative shift.



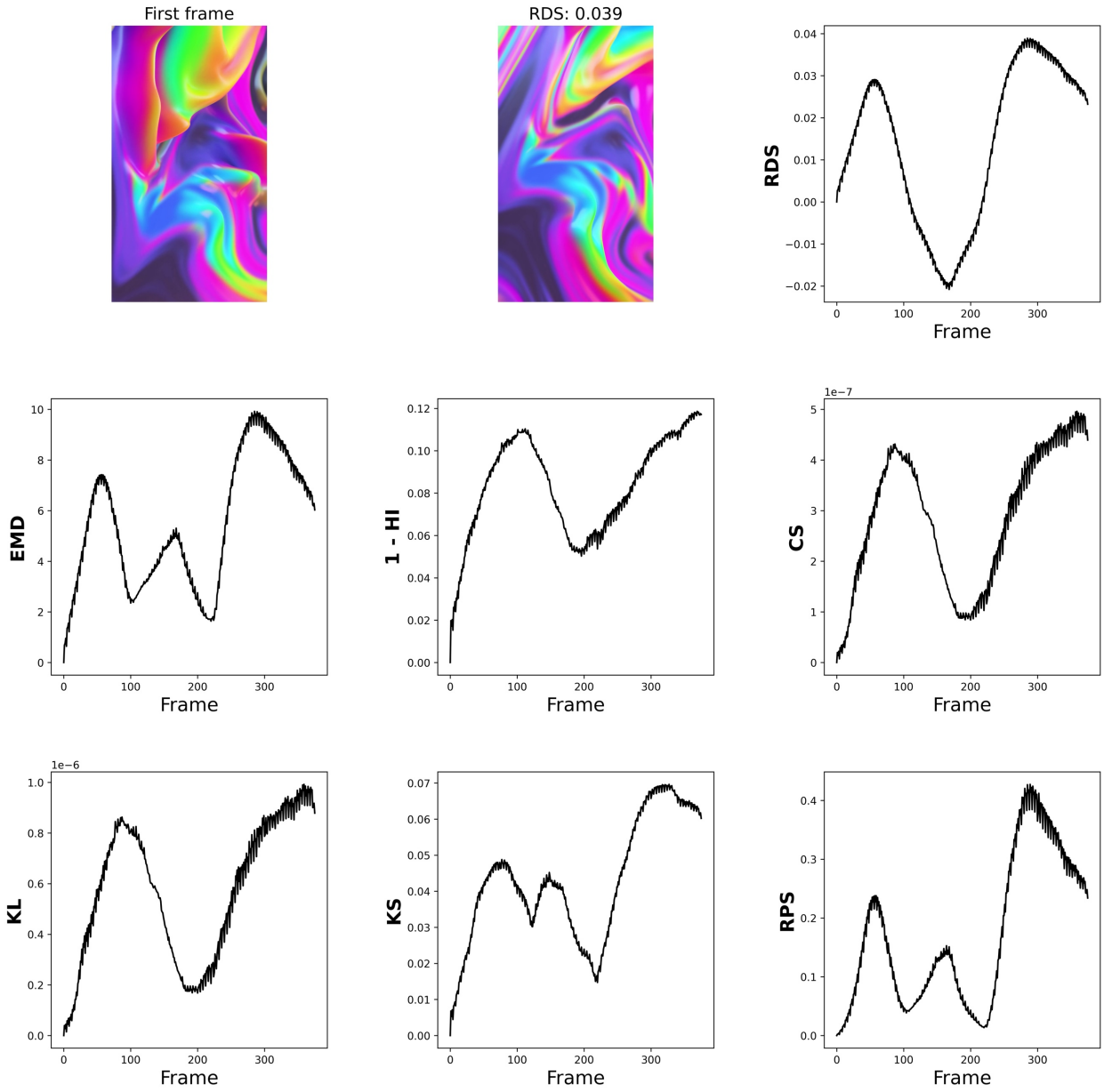
Supplemental figure 25. Detection of signal within a video of moving hexagons. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS produces a simpler, more regular signal because it distinguishes positive and negative shift.



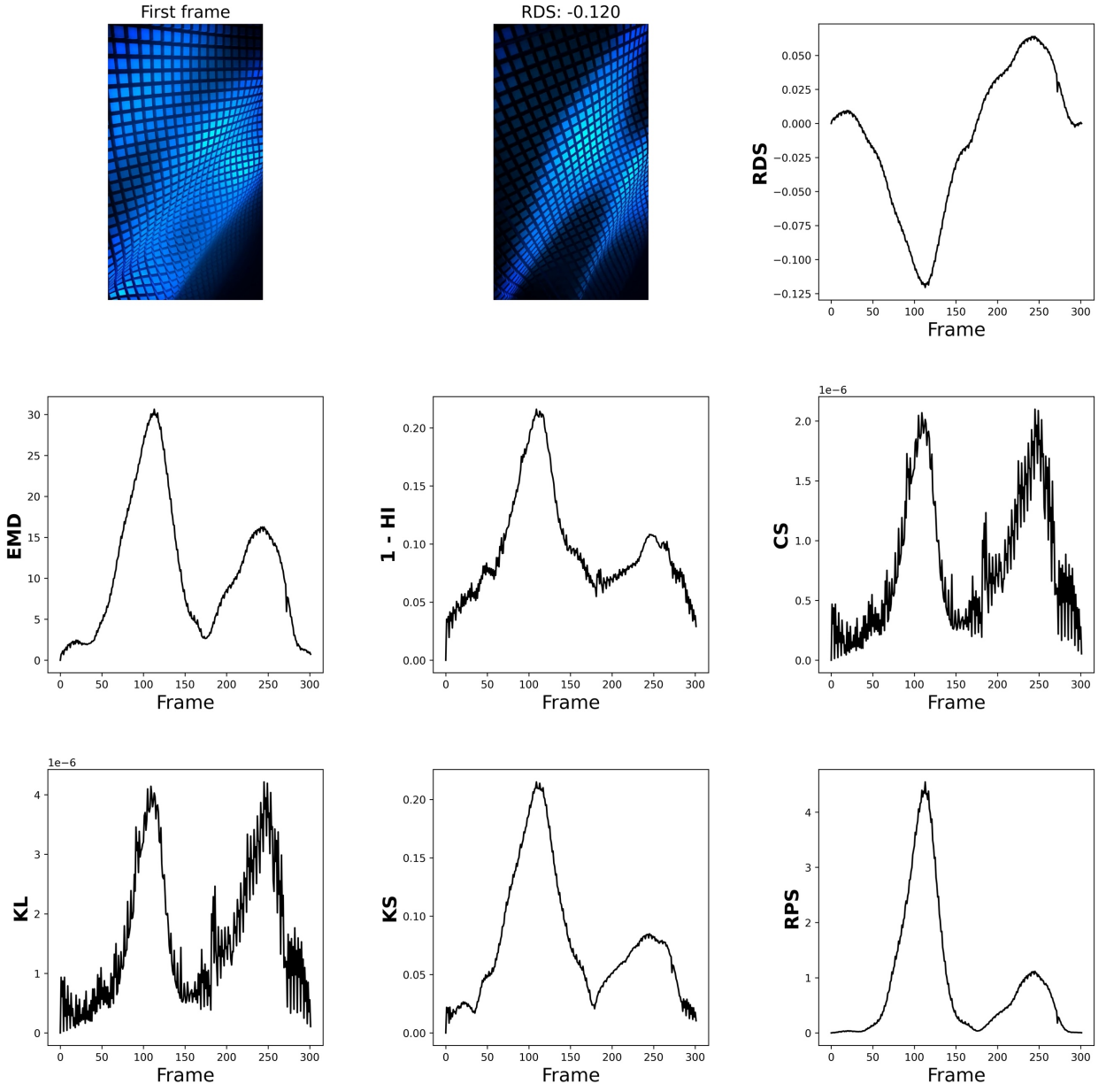
Supplemental figure 26. Detection of signal within a video of moving circles and small line segements. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS produces a simpler, more regular signal because it distinguishes positive and negative shift.



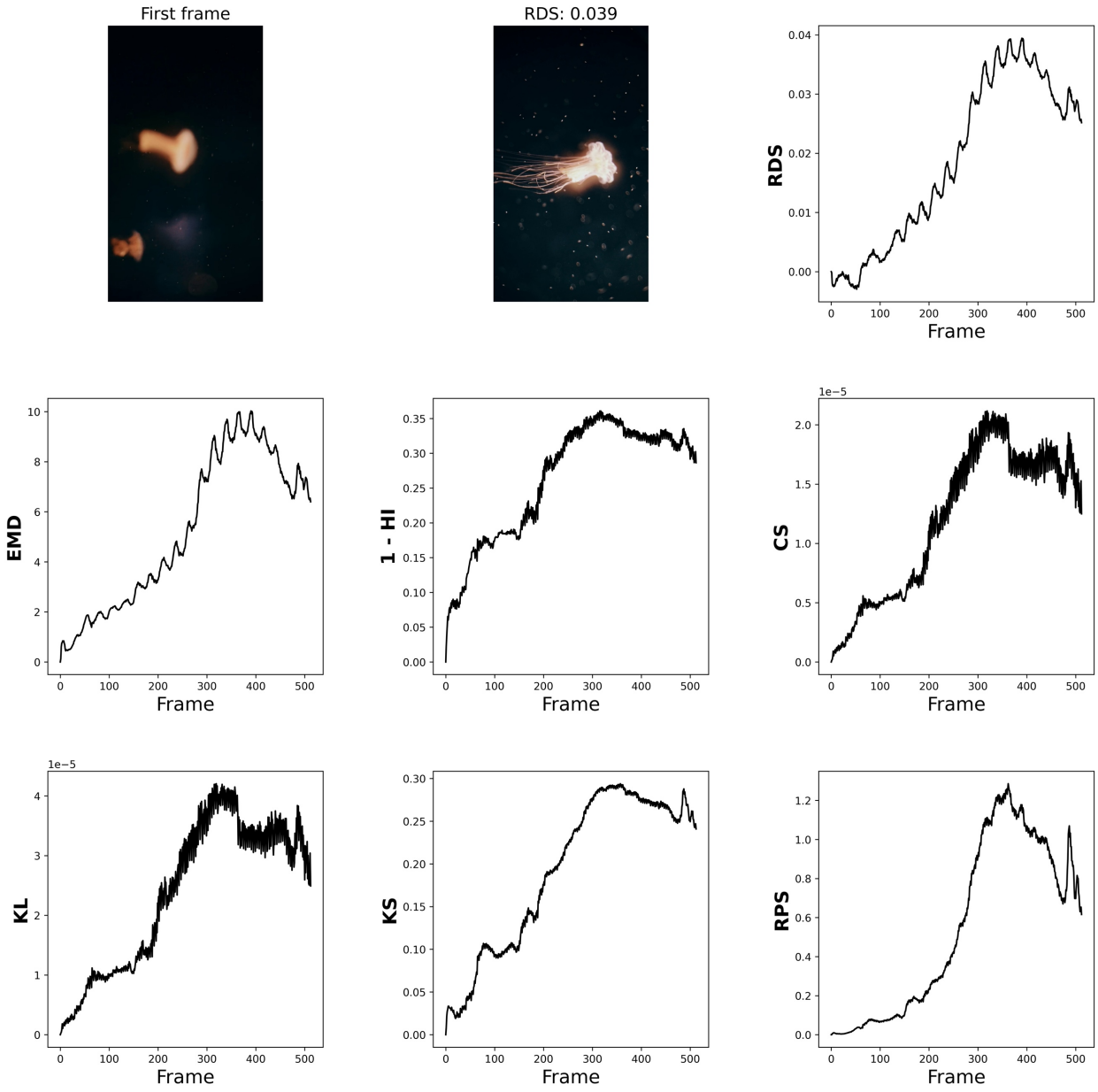
Supplemental figure 27. Detection of signal within a video of amorphous color waves. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS produces a simpler, more regular signal than EMD, KS, and RPS because it distinguishes positive and negative shift.



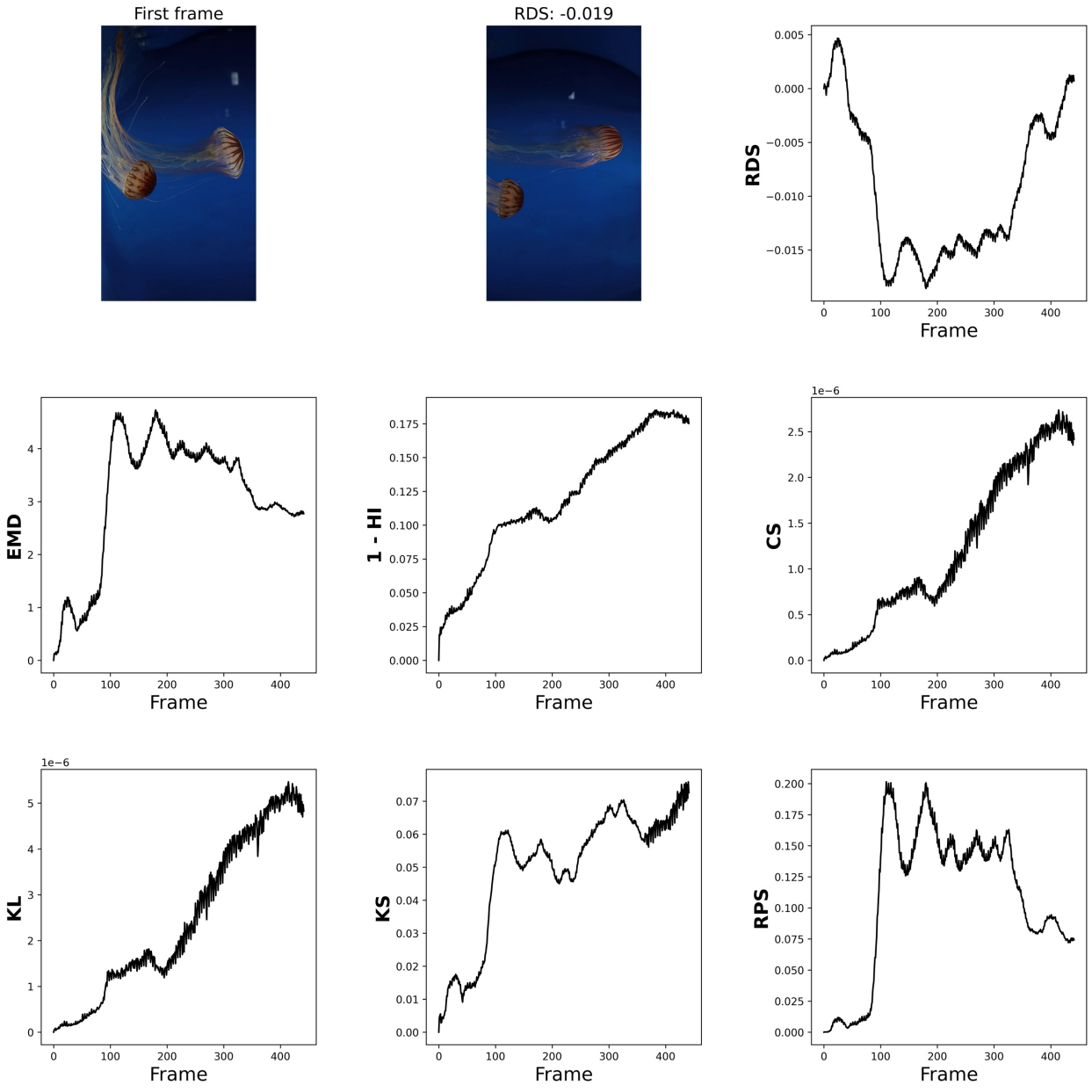
Supplemental figure 28. Detection of signal within a video of a wavy two-dimensional plane of rectangles. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS produces a simpler, sinusoidal signal because it distinguishes positive and negative shift.



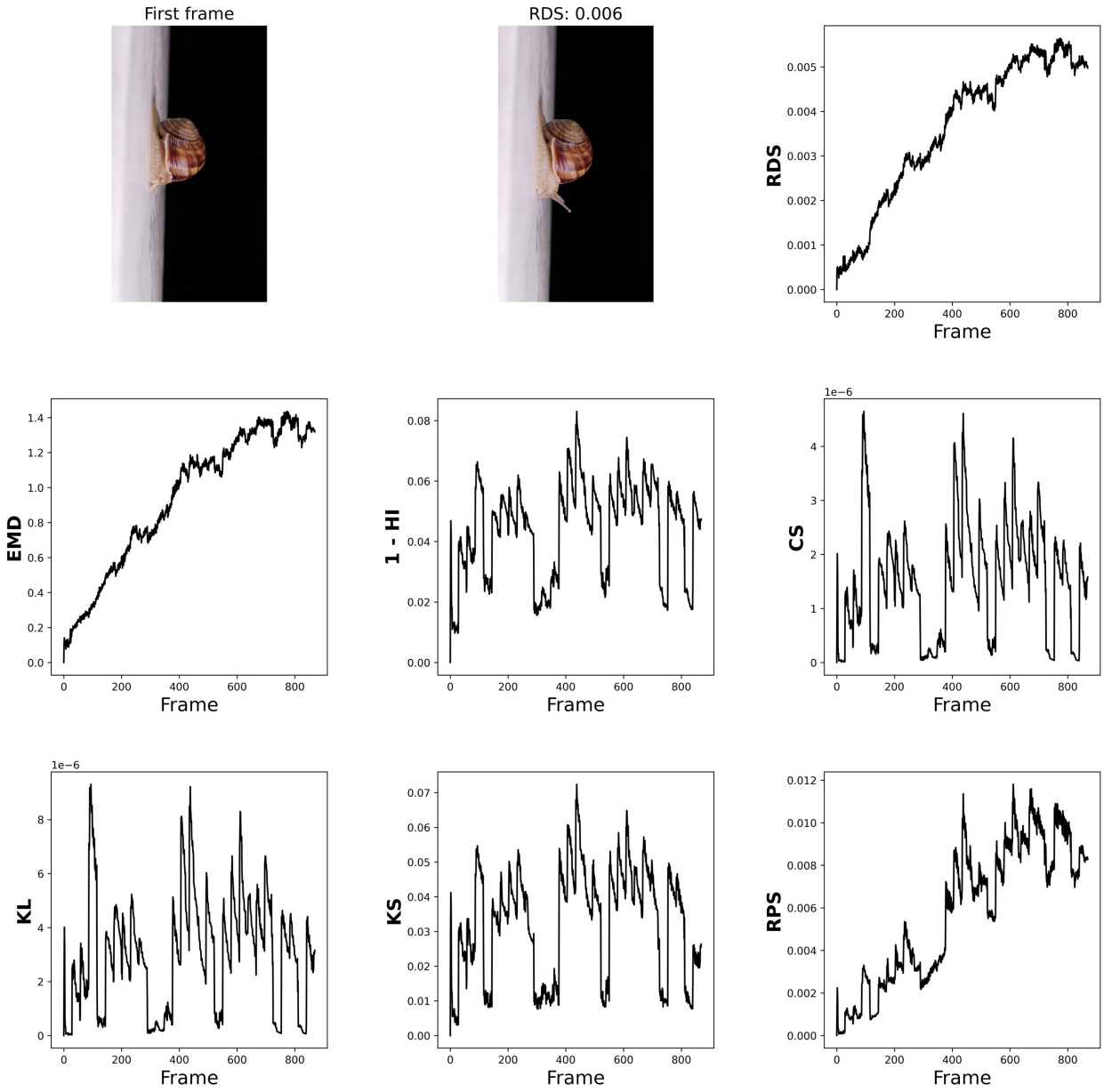
Supplemental figure 29. Detection of undulating jellyfish movement. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS and EMD capture the undulating movement while other measures do not.



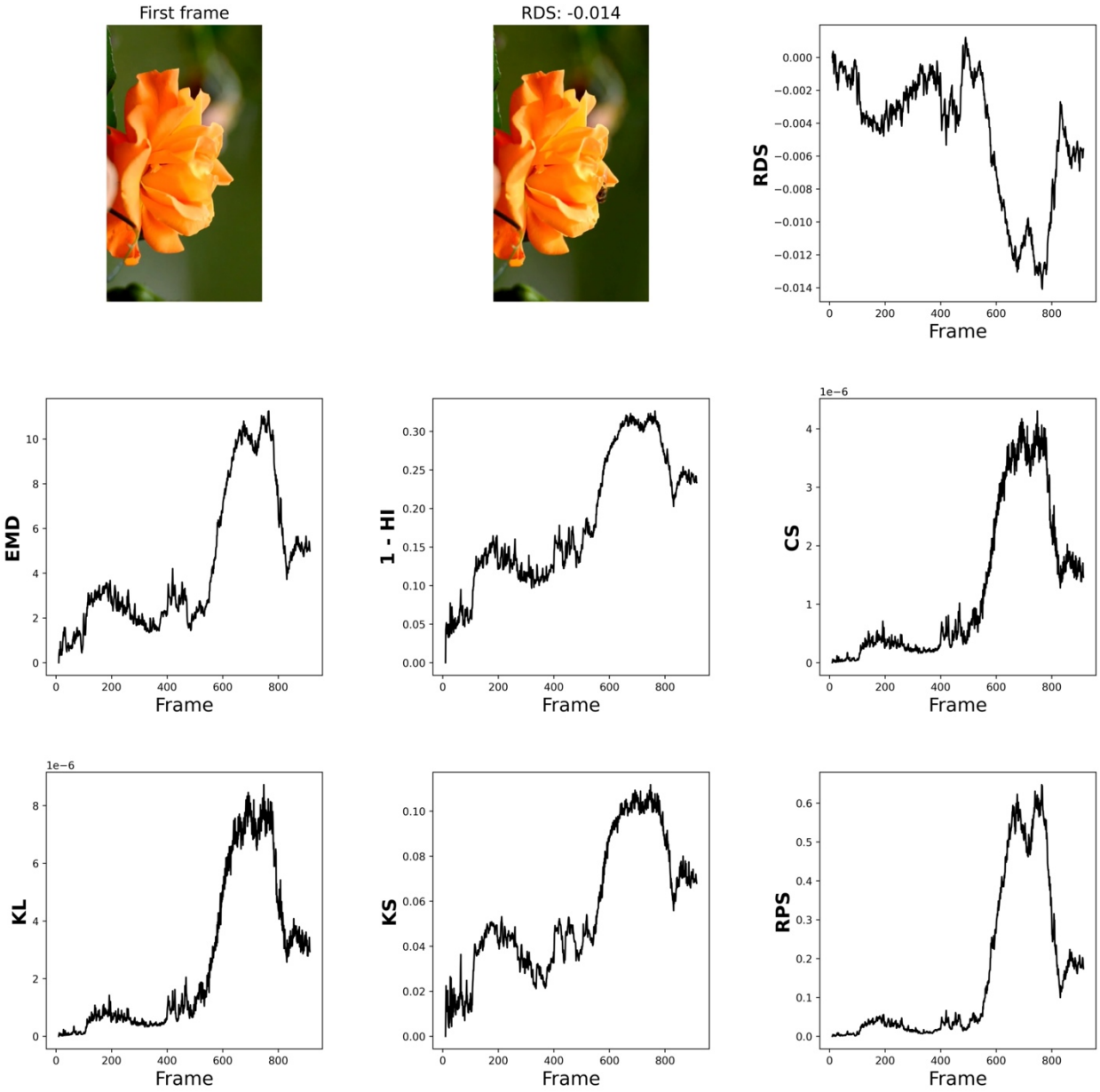
Supplemental figure 30. Detection of undulating jellyfish movement. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. CS, 1 – HI, and KL fail to capture the undulating movement.



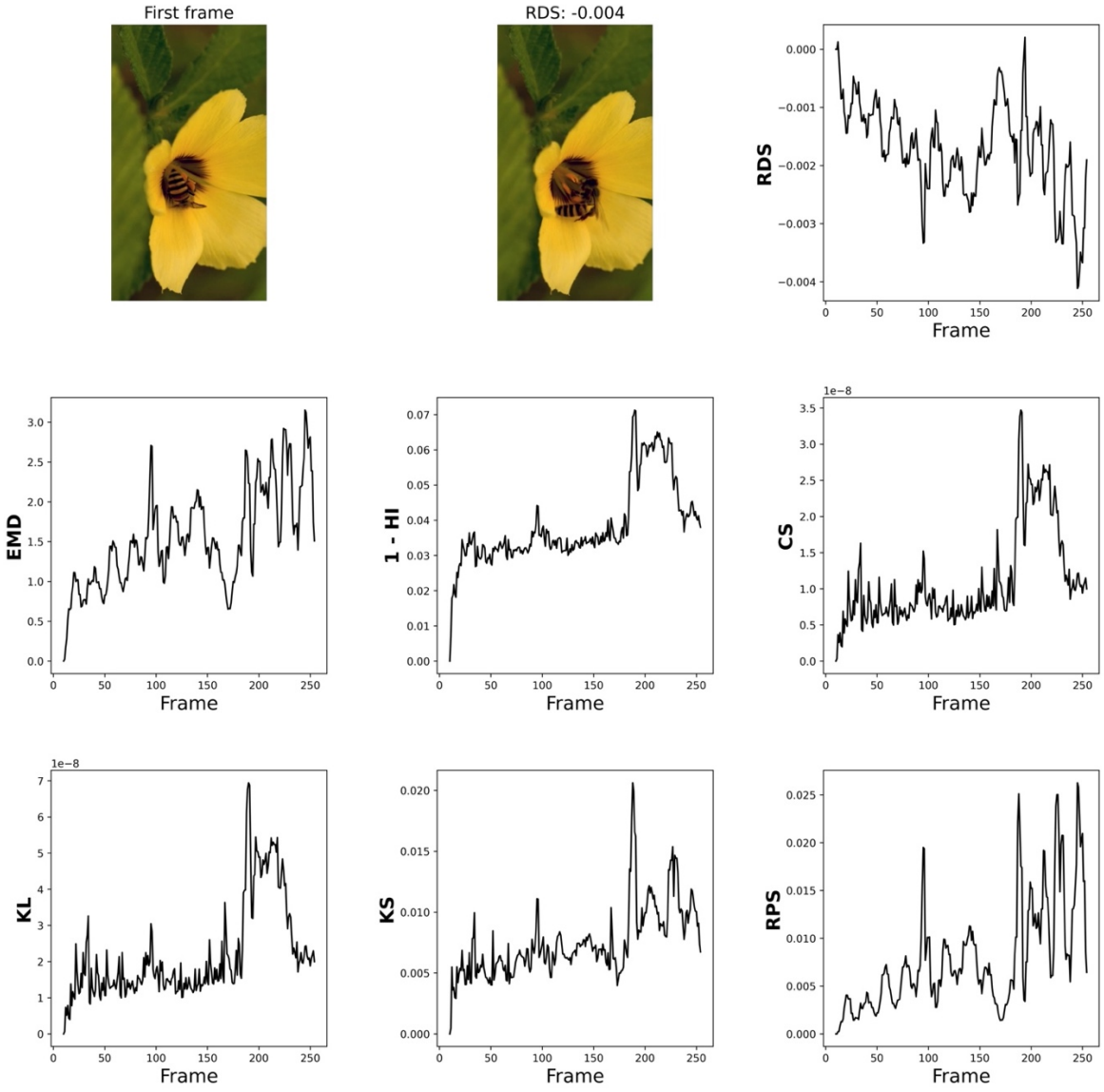
Supplemental figure 31. Detection of a slowly emerging snail. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS, EMD, and to a lesser extent RPS, capture the increased exposure of the snail's light-colored body against the darker background.



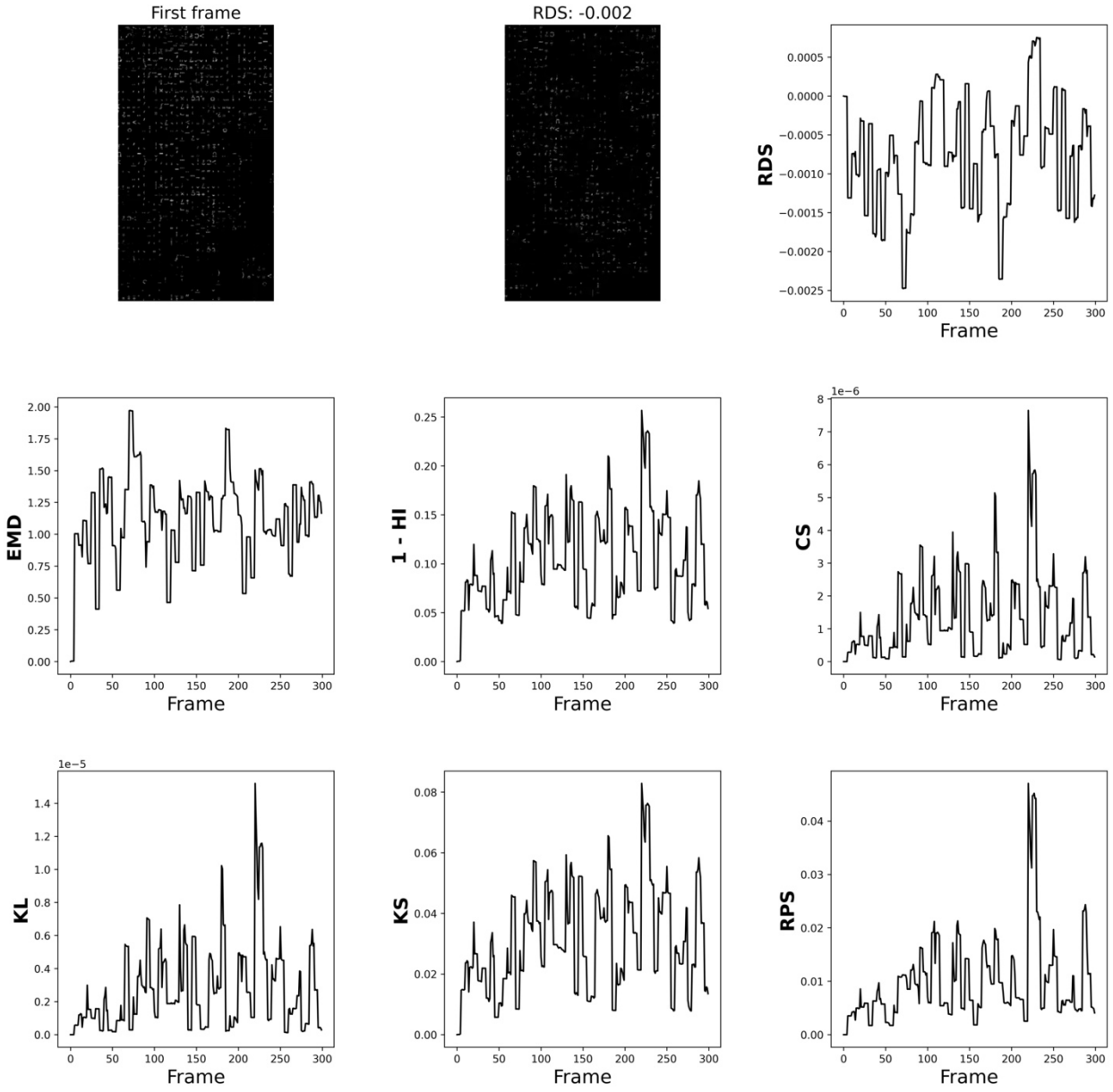
Supplemental figure 32. Video detection of the activity of two bees on a flower. Top right: The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. All measures signal the appearance and activity of two bees (ca. frames 100 to 350) and a large change occurring between frames 600 and 800. This latter change was caused by a substantial decrease in ambient light, which RDS correctly registered as red-shift.



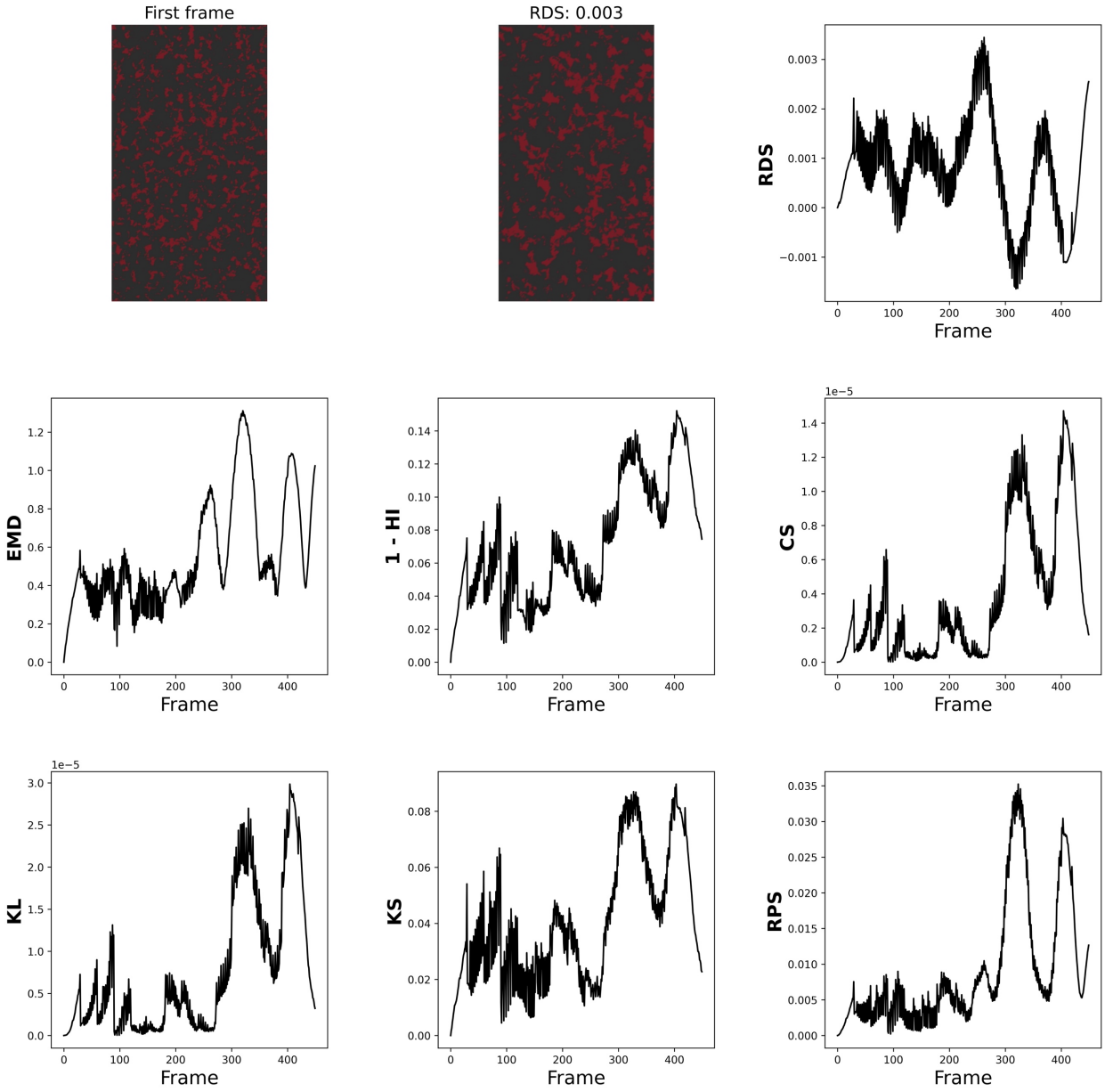
Supplemental figure 33. Video detection of the activity of a wasp on a flower. Top right: The first frame of the video, used as the reference frame. Top center: Frame with the greatest absolute RDS value. Only RDS, EMD, and RPS clearly capture the signal of a black ant quickly crawling across a leaf (frames 80 – 100) and the wasp's activity focused near the center of the flower (frames 160 – 180). All measures register a signal of the wasp leaving the flower (frames 180 – 190). Only 1 – HI, CS, and KL register a clear signal of the wasp's temporary absence (frames 190 – 230).



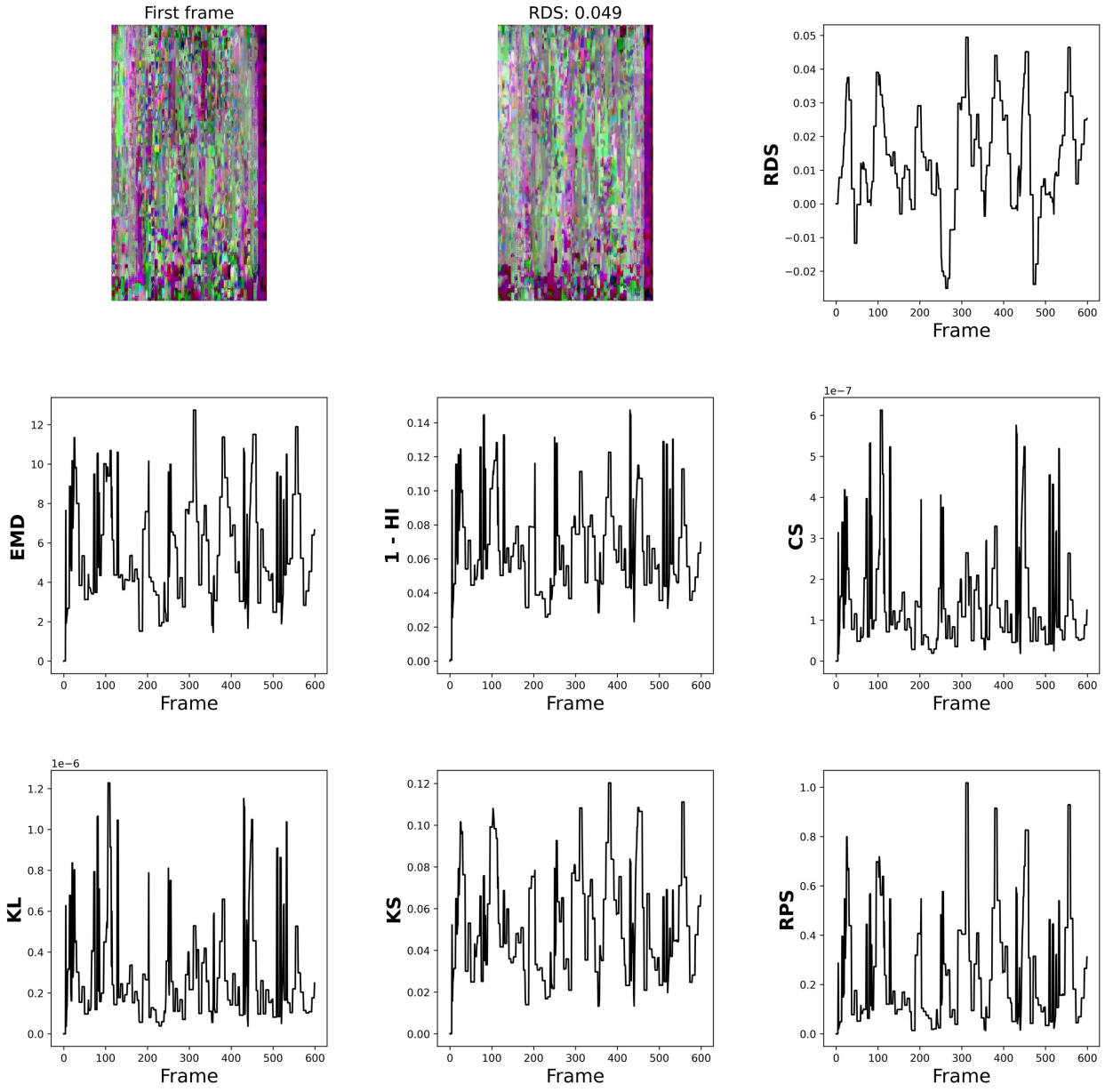
Supplemental figure 34. Detection of signal within a highly pixelated low-contrast video wherein seemingly random clusters of pixels change in seemingly random fashion. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. The signal produced by RDS reveals two and a half repetitions of a complex pattern. EMD reveals a less regular pattern. Other measures suggest little-to-no pattern.



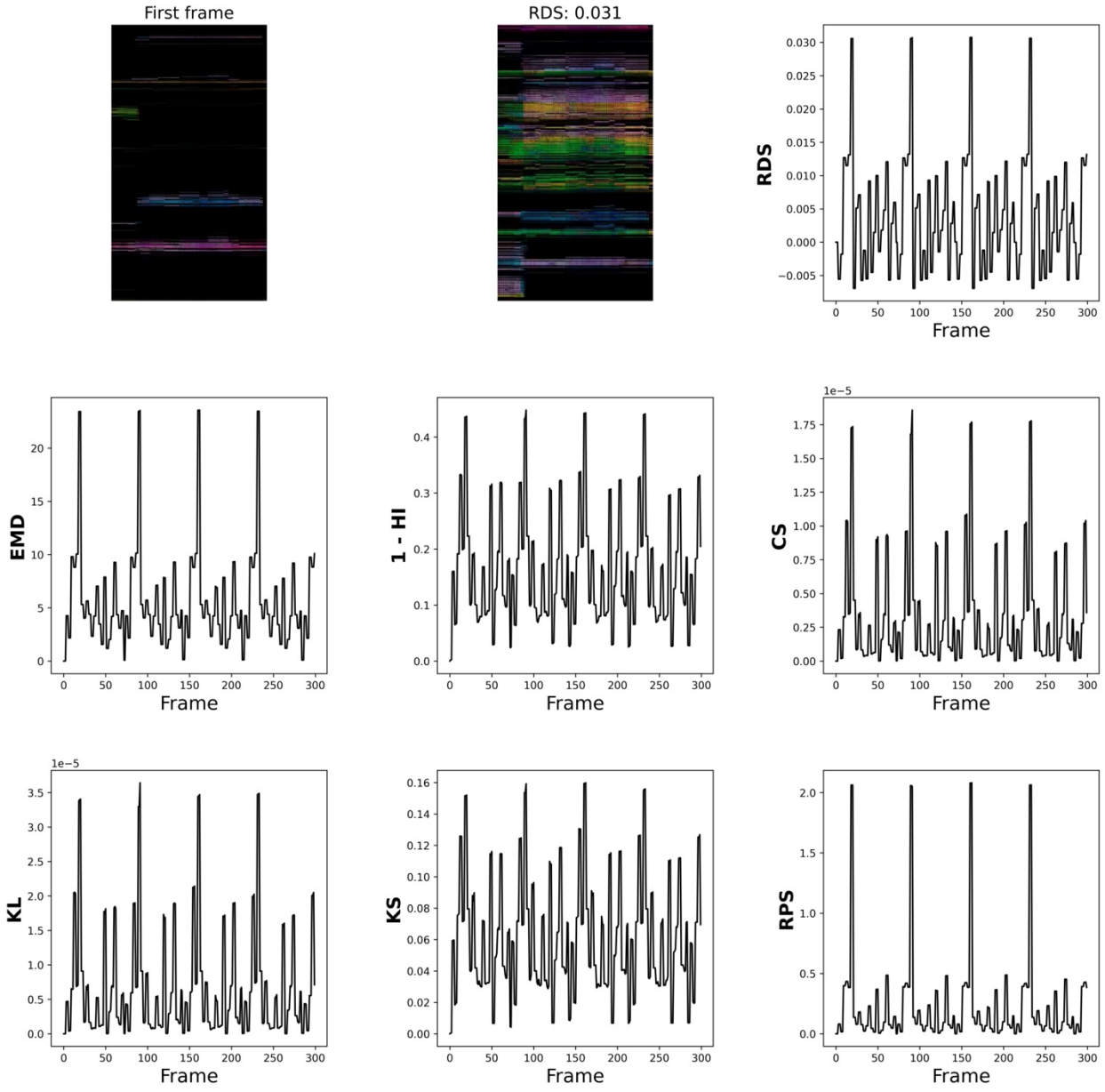
Supplemental figure 35. Detection of signal within a low contrast video wherein a seemingly fractal-like field of amorphous shapes change in seemingly random fashion. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS produces a simpler, more sinusoidal signal, in part, because it distinguishes positive and negative shift.



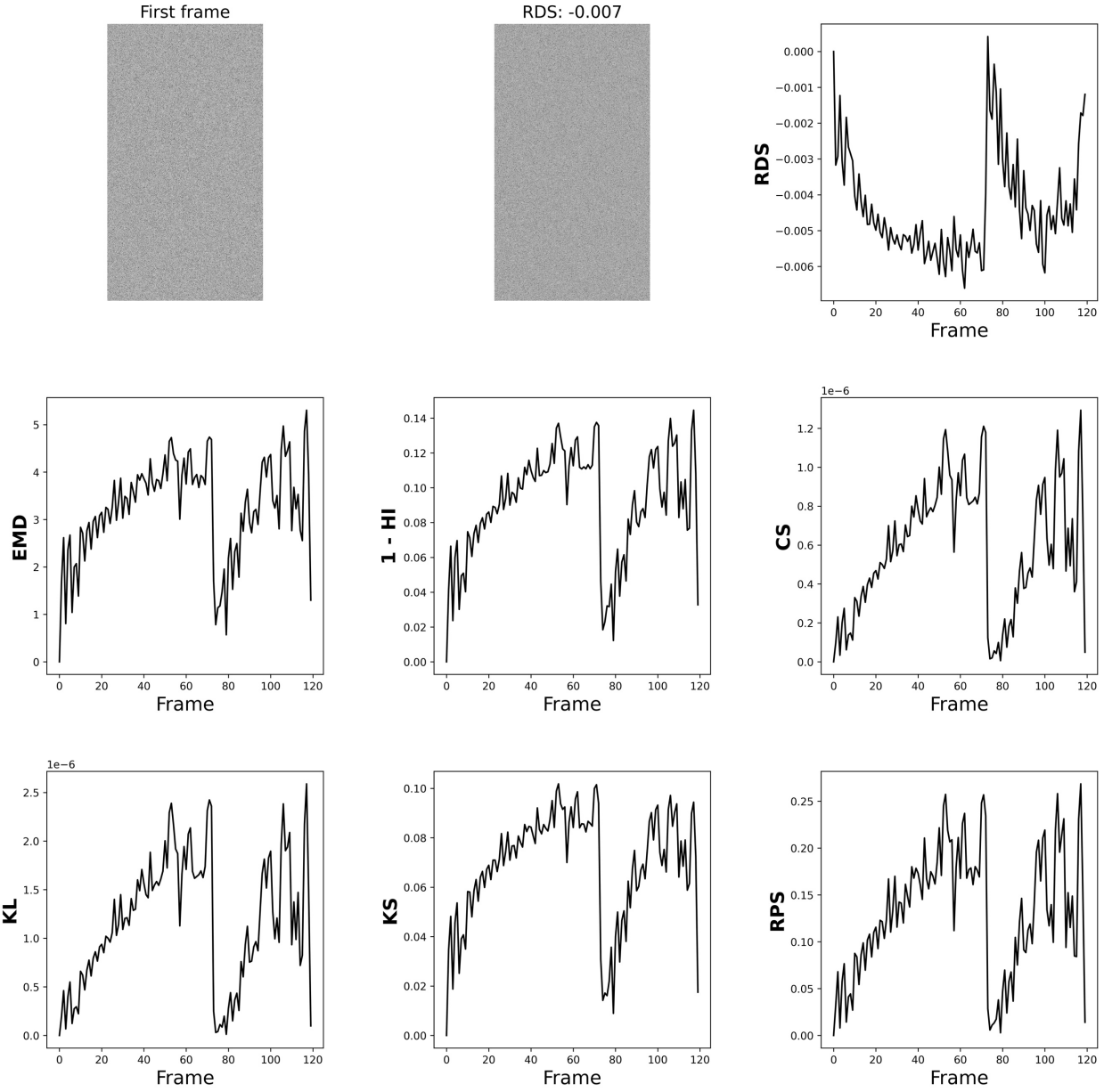
Supplemental figure 36. Detection of signal within a highly pixelated low-contrast video wherein seemingly random clusters of pixels change in seemingly random fashion. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. The signal produced by RDS reveals two a repetition of a complex pattern (three pronounced peaks and a pronounced trough). Other measures suggest little-to-no pattern.



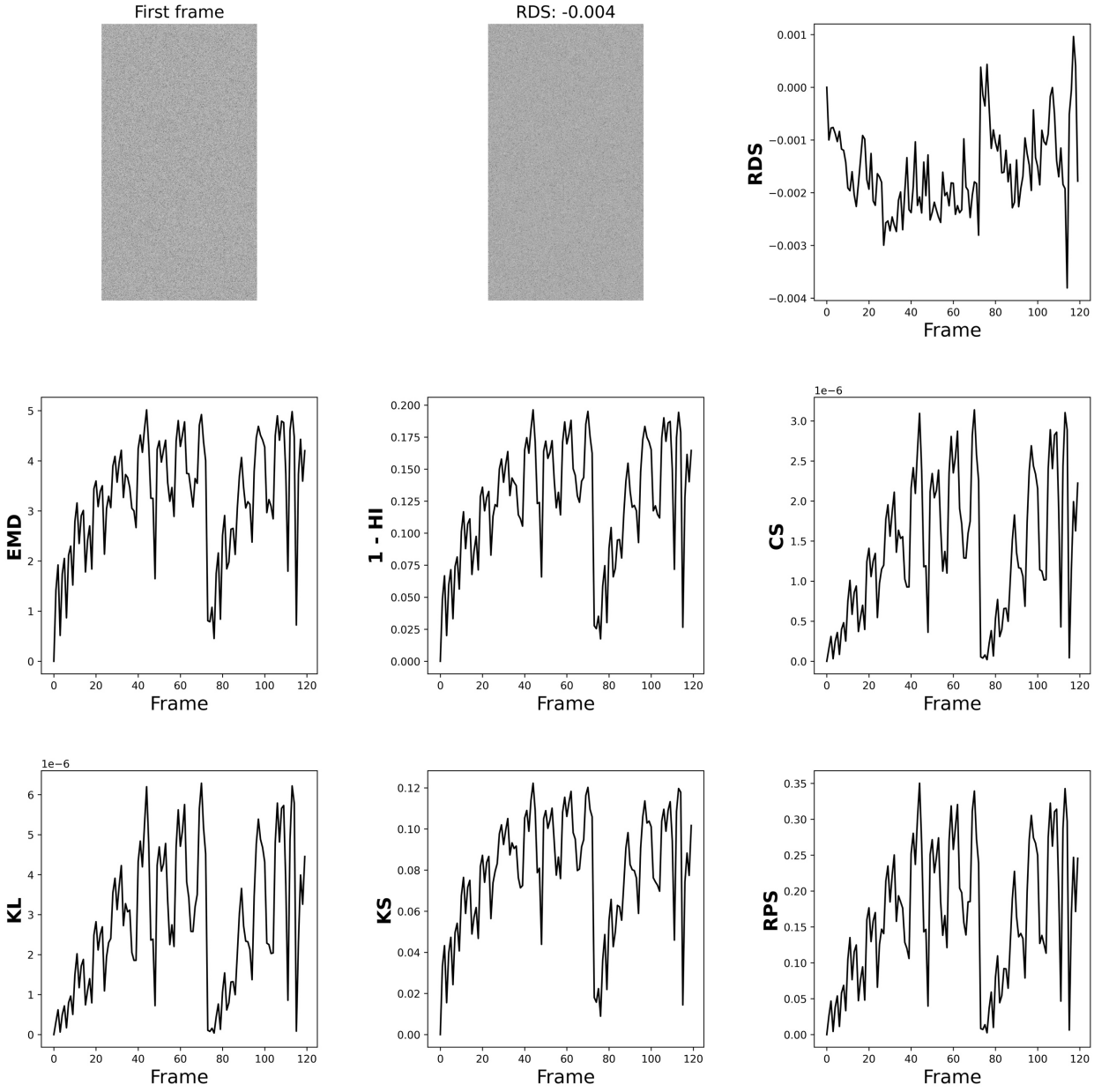
Supplemental figure 37. Detection of signal within a highly pixelated video wherein seemingly random clusters of colored bands change in seemingly random fashion against a black background. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value, i.e., greatest difference from the first frame as measured via RDS. All measures detect a regularly repeating signal.



Supplemental figure 38. Detection of signal within a video of greyscale pixels distributed in seemingly uniform random fashion. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. Each measure captures a similar pattern, which is reversed for RDS because it distinguishes positive and negative shift.



Supplemental figure 39. Detection of signal within a video of greyscale pixels distributed in seemingly uniform random fashion. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value. RDS fails to capture the pattern revealed by other measures, which are similar to the patterns in Supplemental figure 38.



Supplemental figure 40. Detection of signal within a highly pixelated greyscale video of apparent white noise. Top right: First frame of the video, used as a reference frame. Top center: Frame with the greatest absolute RDS value, i.e., greatest difference from the first frame as measured via RDS. All measures except RDS reveal a distinct pattern.